

# ADVANS JAVA

By:

Nagore Babu. Durga Soft



Q. What are the differences between Classes, Abstract classes and Interfaces?

⇒ 1) Classes are able to allow only concrete methods.

Abstract classes are able to allow both concrete methods and abstract methods.

Interfaces are able to allow only abstract methods.

2) For classes only we are able to create both reference variables and Objects.

For abstract classes and interfaces, we are able to create reference variables and we are unable to create Objects.

3) In case of interfaces, by default, all the variables are "public static final".

No default cases are existed for variables in the case of classes and abstract classes.

4) In case of interfaces, by default all the methods are "public abstract".

No default cases are existed for the methods in the case of classes and abstract classes.

5) Constructors are possible in classes and abstract classes.

Constructors are not possible in interfaces.

6) Classes will provide less sharability.

Abstract classes will provide middle level sharability.

Interfaces will provide more sharability.

### Procedure to write classes:-

1) Declare a class by using "Class" keyword.

2) Declare instance variables as per the requirement.

- 3) Provide initializations to instance variables by using parameterized constructor.
- 4) In main class, in main() method, create object for the respective class by accessing parameterized constructor.
- 5) Access the required methods of the respective class.

```
class Employee
{
```

```
    String eid;
    String ename;
    String eaddr;
    float esal;
```

```
Employee(String eid, String ename, float esal, String eaddr)
```

```
    {
        this.eid = eid;
        this.ename = ename;
        this.esal = esal;
        this.eaddr = eaddr;
    }
```

```
public void getEmployeeDetails()
```

```
    System.out.println ("Employee Details");
    System.out.println ("-----");
    System.out.println ("Employee Id :" + eid);
    System.out.println ("Employee Name :" + ename);
    System.out.println ("Employee Salary :" + esal);
    System.out.println ("Employee Address :" + eaddr);
```

## Class Test

```

{
    PSVM (String []args)
}

Employee emp = new employee ("E-11", "Durga", 50000.0f, "Hyd");
emp.getEmployeeDetails ();
}

```

Dynamic Input in java:-

There are three approaches to get dynamic input in java.

1) BufferedReader

2) Scanner

3) Console

[Note:- Dynamic input means the input which is putted manually in Command prompt.]

1) BufferedReader:-

If we want to take dynamic input in java applications by using BufferedReader then we have to use the following construction.

`BufferedReader br = new BufferedReader(new InputStreamReader(System.in));`

Where 'in' is a predefined static reference variable, it is used to refer a predefined InputStream object connected with Command prompt in order to read data from command prompt.

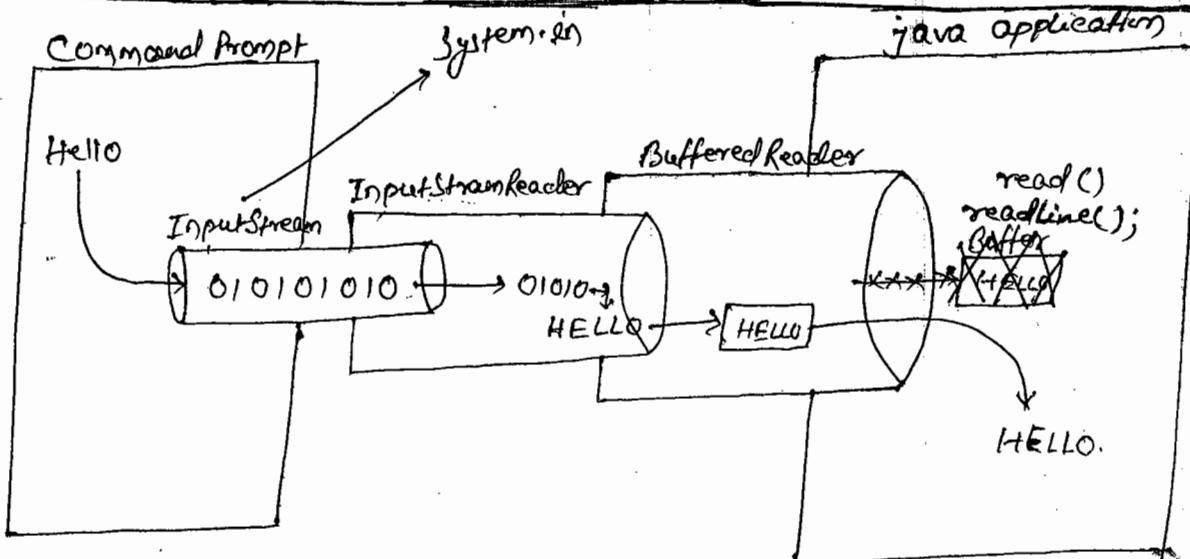
When we provide data on command prompt, data will be transferred to InputStream object in the form of bytes representation (Binary representation).

Where "InputStreamReader" can be used to convert data from bytes representation to character representation.

In the above context, we can read data directly from InputStreamReader, but the performance of the java application will be reduced.

Here, to improve java application performance we have to use BufferedReader:

```
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
```



If we want to read data from BufferedReader then we have to use the following two methods.

- 1) `readLine()` ✓
- 2) `read()` ✓

Q. What is the difference between `read()` method and `readLine()` method?

→ `readLine()` method can be used to read a line of text data from command prompt and it will return that line of text data in the form of String.

```
public String readLine()
```

→ `read()` method can be used to read a single character from command prompt and it will return that character in the form of its ASCII value.

(3)

public int read()

Ex:-①

import java.io.\*;

Class Test

{

PSVM (String [] args) throws Exception

{

BufferedReader br = new BufferedReader(new InputStreamReader  
(System.in));

S.O.P("Enter Text Data :"); → Durga Software Solutions

String data = br.readLine();

S.O.P("Enter the same data again :"); → Durga Software Solutions

int val = br.read();

S.O.P("First Entered :" + data); → Durga Software Solutions

S.O.P("Second Entered :" + val + "[" + (char) val + "]"); → 68[D]

{}

Ex:-② import java.io.\*;

Class Test

{

PSVM (String [] args) throws Exception

{

BufferedReader br = new BufferedReader(new InputStreamReader  
(System.in));

S.O.P("First Value :"); → 10

String fval = br.readLine();

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```

S.O.P("Second Value :"); → 20
String sval = br.readLine();
int val1 = Integer.parseInt(fval);
int val2 = Integer.parseInt(sval);

S.O.P("Addition :" + (val1 + val2)); → 30
S.O.P("Subtraction :" + (val1 - val2)); → -10
S.O.P("Multiplication :" + (val1 * val2)); → 200
}
}

```

To take numeric data as dynamic input if we use BufferedReader approach then we have to use explicitly wrapper classes in order to convert data from String type to the respective primitive types.

[Note:- In case of BufferedReader, if we enter numeric data as dynamic input then br.readLine() method will take that numeric data in the form of string, where we have to convert numeric data from String type to the respective primitive type, for this we have to use wrapper classes.



## Scanner:-

In java applications, if we want to take dynamic input without using wrapper classes then we have to use scanner.

- Scanner is a class provided by JDK 5.0 version in the form of java.util.Scanner.
- If we want to use scanner class in java applications then we have to use the following class. Steps.
  - i) Create Scanner class Object
  - ii) Get dynamic input from command prompt.

### 1) Create Scanner class Object:-

To create scanner class object we have to use the following constructor from java.util.Scanner class.

public Scanner (InputStream is)
---------------------------------

### 2) Get dynamic input from Command prompt:-

To read String data as dynamic input we have to use the following methods.

public String next()
public String nextLine()

To get primitive data as dynamic input then we have to use the following methods

public xxx nextXXX()
----------------------

Where xxx may be byte, short, int, float, .....

```
import java.util.*;  
class Test  
{  
    public static void main(String[] args)  
    {  
        Scanner s = new Scanner(System.in);  
        System.out.print("Employee Number :");  
        int eno = s.nextInt();  
        System.out.print("Employee Name :");  
        String ename =  
            s.nextLine();  
        System.out.print("Employee Salary :");  
        float esal = s.nextFloat();  
        System.out.print("Employee Address :");  
        String eaddr = s.nextLine();  
        System.out.println("Employee Details");  
        System.out.println("-----");  
        System.out.println("Employee Number : " + eno);  
        System.out.println("Employee Name : " + ename);  
        System.out.println("Employee Salary : " + esal);  
        System.out.println("Employee Address : " + eaddr);  
    }  
}
```

Q What is the difference between `next()` method and `nextLine()` method?

→ `next()` method can be used to read only one word (first word) from the provided dynamic input.

`nextLine()` method can be used to read the complete line of text which we entered as dynamic input on command prompt.

Ex:-

```
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter Text Data :");
        String data1 = s.nextLine();
        System.out.print("Enter the Data again :");
        String data2 = s.next();
        System.out.println("nextLine()   Op: "+data1);
        System.out.println("next()       Op: "+data2);
    }
}
```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### ③ Console:-

To take dynamic input in java applications if we use BufferedReader and Scanner approaches then we are able to get the following drawbacks.

- ① For every single dynamic input we have to use two instructions explicitly, it will increase no. of instructions in java programs.
  - a) System.out.println(--) method to display request message on command prompt.
  - b) readLine() or next() or nextXXX() methods to read dynamic input.
- ② No Security for the dynamic input like password data, PIN numbers, ...

To Overcome the above problems we have to use java.io.Console class.

If we want to use console class in java applications then we have to use the following steps.

#### ① Create console class Object:-

To get console class object we have to use the following method from java.lang.System class.

`public static Console console()`

(6)

## ② Read dynamic input through Console:-

To display a message and to read data in the form of String we have to use the following method.

**public String readLine(String message)**

To display a message and to read password data without displaying on command prompt we have to use the following method.

**public char[] readPassword(String message)**

Note:- If we want to take primitive data as dynamic input by using console then we have to use wrapper classes.

~~E~~

~~Ex:-~~

```

import java.io.*;
Class Test
{
    public void main(String[] args) throws Exception
    {
        Console c=System.console();
        String uname=c.readLine("User Name :");
        char[] pwd=c.readPassword("Password :");
        String upwd=new String(pwd);
        if(uname.equals("durga") & upwd.equals("durga"))
        {
            System.out.println("Congratulations User !");
            System.out.println("U R login is Success");
        }
    }
}
```

28-08-2015

```
}
```

else

```
{
```

```
    S.O.Pln("Sorry User!);
```

```
    S.O.Pln("U R Login is Failure");
```

```
    S.O.Pln("Use Ur User Name and password");
```

```
}
```

O/P:-

User Name:

password :

Sorry User!

U R Login is Failure

Use Ur User Name and password.



## \* FileInputStream and FileOutputStream :-

### 1) FileOutputStream :-

FileOutputStream is a byte Oriented Stream, it will carry data from java application to a particular target file in the form of bytes.

To carry data from java application to a particular file by using FileOutputStream then we have to use the following steps.



### 1) Create FileOutputStream:-

To create FileOutputStream class object, we have to use the following constructor.

```
public FileOutputStream(String file-name)
```

- It will override existed data with new data at each and every write operation.

```
public FileOutputStream(String file-name, boolean b)
```

- It will append new data with the existed data in the specified file if 'b' value is true.
- If 'b' value is false then it will override existed data with new data.

Ex:-

```
FileOutputStream fos = new FileOutputStream("abc.txt", true);
```

When JVM encounter the above instruction, JVM will perform the following actions.

- 1) JVM will check whether the specified file is existed or not at the respective location.
- 2) If the specified file is existed then JVM will prepare FileOutputStream.
- 3) If the specified file is not existed then JVM will create a file with the specified name and JVM will create FileOutputStream.

2) Declare the data which want to send:-

```
String data = new String("Hello");
```

3) Convert data from String type to byte[] type:-

To convert data from String type to byte[] type we have to use the following method:

```
public byte[] getBytes()
```

Ex:-

```
byte[] b = data.getBytes();
```

4) Write byte[] data to FileOutputStream:-

To write byte[] data to FileOutputStream we have to use the following method.

```
public void write(byte[] b) throws Exception.
```

Ex:- fos.write(b);

5) Close FileOutputStream:-

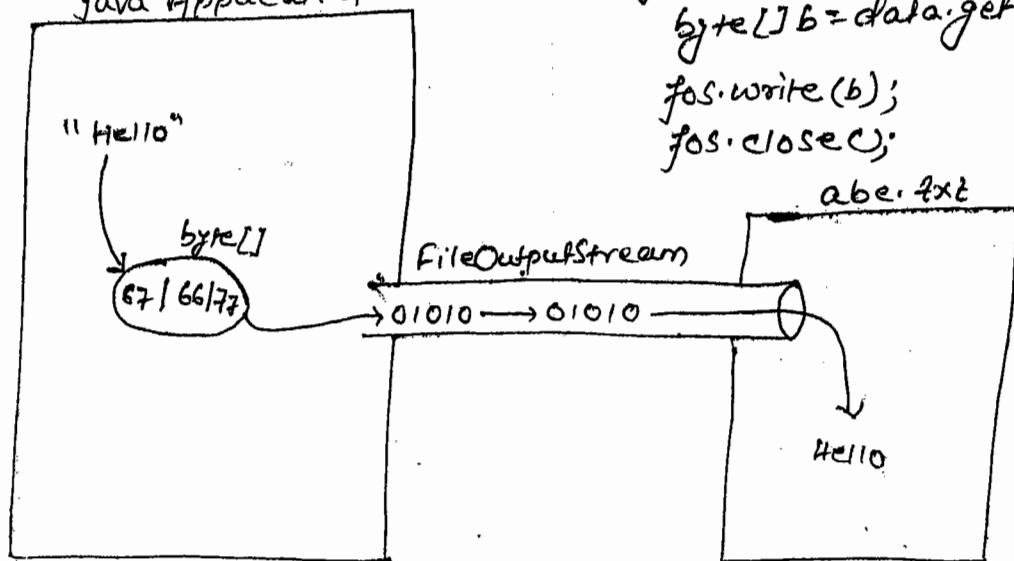
To close FileOutputStream we have to use the following method.

```
public void close() throws IOException.
```

Ex:- fos.close();

```
FileOutputStream fos = new FileOutputStream ("abc.txt", true);
```

java Application.



```
String data = "Hello";
```

```
byte[] b = data.getBytes();
```

```
fos.write(b);
```

```
fos.close();
```

abc.txt

Hello

Ex:-

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

```
Class Test
```

{

```
PSVM (String [] args) throws Exception
```

```
FileOutputStream fos = new FileOutputStream ("abc.txt", true);
```

```
String data = new String ("Hyderabad");
```

```
byte[] b = data.getBytes();
```

```
fos.write(b);
```

```
fos.close();
```

}

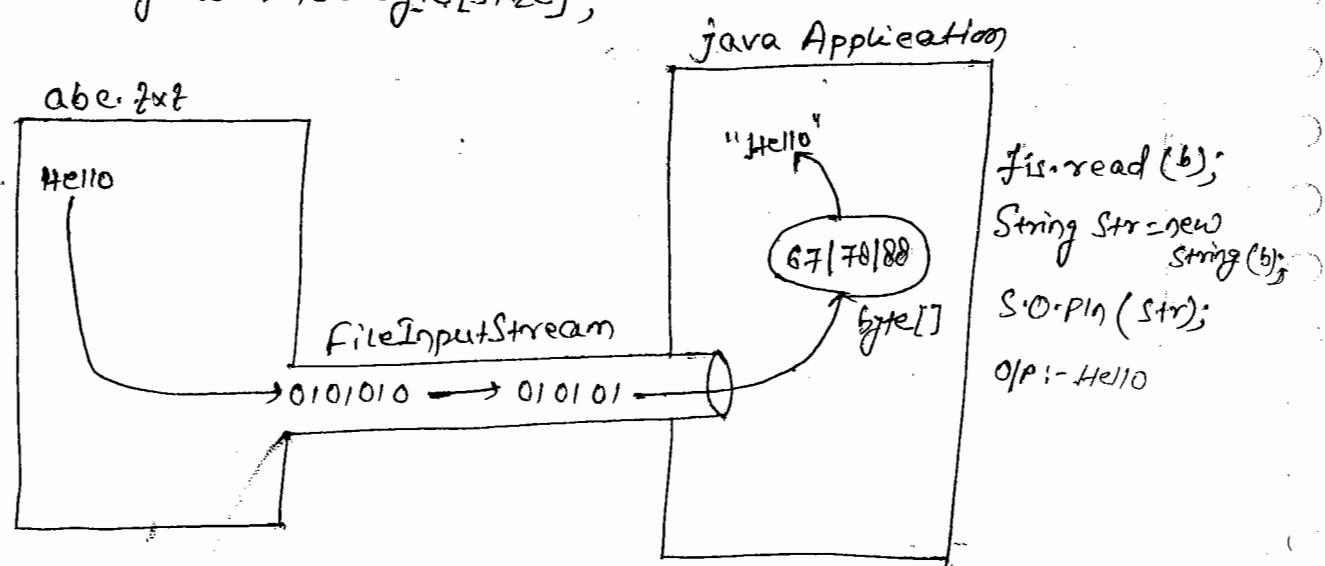
}

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## FileInputStream:-

It can be used to carry data from a particular source to java applications.

```
fileInputStream fis = new FileInputStream ("abc.txt");
int size = fis.available();
byte[] b = new byte [size];
```



If we want to get data from a particular source file to java application by using FileInputStream then we have to use the following steps.

1) Create FileInputStream class Object :-

→ To create ~~file~~ FileInputStream class object, we have to use the following constructor.

```
public FileInputStream (String file-name) throws FileNotFoundException
```

~~Ex:-~~ FileInputStream fis = new FileInputStream ("abc.txt");

→ When JVM encounter the above instruction, JVM perform the following actions.

(9)

- 1) JVM will take the provided file name and search for it at the specified location.
- 2) If the specified file is not available at the specified location then JVM will rise an exception like "java.io.FileNotFoundException".
- 3) If the specified file is existed then JVM will create FileInputStream and JVM will send the complete data from the specified file to FileInputStream in bytes representation.
- 2) Identify data size from FileInputStream:-

public int available()

Ex:- int size = fis.available();

- 3) Create byte[] with the data size:-

byte[] b = new byte [size];

- 4) Read data from FileInputStream to byte[]:-

To read data from FileInputStream to byte[] we have to use the following method.

public void read (byte[] b)

Ex:- fis.read(b);

String data = new String(b);

System.out.println(data);

## 5) close FileInputStream:-

fis.close();

Ex:-

```
import java.io.*;
Class Test
{
    public void main (String [ ] args) throws Exception
    {
        FileInputStream fis = new FileInputStream ("abc.txt");
        int size = fis.available ();
        byte [ ] b = new byte [size];
        fis.read (b);
        String data = new String (b);
        System.out.println (data);
        fis.close ();
    }
}
```

### Exception Handling:-

Exception is runtime error causes abnormal termination to the applications.

OR

Exception is an unexpected event occurred at runtime provided by the developers while entering dynamic input to the java applications, provided by the database while executing SQL queries and JDBC applications, provide by the remote machine while establish the connection between local machine and remote machine in distributed applications causes abnormal termination to the applications.

→ There are two types of terminations in java applications.

① Smooth termination:-      ② Abnormal termination:-

② Terminating the program in the middle is called as Abnormal termination.

In java application exceptions will provide abnormal terminations to the applications, where to remove abnormal terminals we have to handle exceptions, for this we have to use 'Exception Handling mechanisms'.

① Terminating the program at the end point of the program is called as Smooth termination.

### throw keyword:-

The main purpose of 'throw' keyword is to rise exceptions explicitly as per the developer's requirement.

### Syntax:-

```
throw new Exception_Name([Param_List]);
```

```
Ex:- throw new IOException("My own IO Exception");
```

→ There are two ways to handle exceptions.

- ① Using throws keyword
- ② Using try-catch-finally

### ① Using throws Keyword:-

"throws" Keyword can be used to bypass the generated exception from present method to caller method in order to handle.

→ "throws" Keyword must be used in method prototype, not in method body.

→ "throws" Keyword is able to allow more than one exception class in method prototype.

```
Void m1() throws ArithmeticException, IOException  
{  
-----  
}
```

Ex:-

```
import java.io.*;  
Class A {  
Void add() throws Exception {  
Concat();  
}  
Void concat() throws IOException  
{  
throw new IOException ("My own IOException");  
}  
Class Test {  
Psvm (String[] args) throws throwable  
A a = new A();  
a.add();  
}
```

02/09/2015

## try-catch-finally :-

In java applications, "throws" keyword is not really exception handler, because, throws keyword is able to bypass the exception from present method to the caller method, it will not handle the exceptions really.

In java applications, if we want to handle the exceptions really then we have to use "try-catch-finally".

Syntax:-

```
try
{
    --- instructions ---
}

Catch(Exception_Name e)
{
    --- instructions ---
}

Finally
{
    --- instructions ---
}
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Where "try" block is able to include a set of instructions may generate an exception. providing instructions in try block is not giving any guarantee to generate exceptions.

If any exception is generated in try block then JVM will bypass flow of execution to catch block by skipping remaining instructions in try block.

If no exception is generated in try block then JVM will skip

flow of execution to finally break directly by skipping catch block.

→ Where the purpose of "catch" block is to catch the exception from try block and display <sup>the</sup> exception details on command prompt or console.

To display exception details, we have to use the following three approaches.

① e.printStackTrace();

→ It able to display the exception details like name of the exception, description of the exception and location of the exception.

② System.out.println(e);

→ It able to display the exception details like name of the exception and description of the exception.

③ System.out.println(e.getMessage());

→ It able to display only description of the exception.

```
class Test
{
    public void psvm(String[] args)
    {
        try
        {
            int i=10;
            int j=0;
            float f=i/j; // here ArithmeticException is raised.
            System.out.println(f);
        }
    }
}
```

```

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

```

Where "finally" block is able to include a set of instructions to execute irrespective of getting exception in try block and irrespective of executing catch block.

```

class Test
{
    public String[] args)
    {
        System.out.println ("Before try");
        try
        {
            System.out.println ("Inside try, before exception");
            int i=10;
            int j=0;
            float f = i/j;
            System.out.println ("Inside try, after exception");
        }
    }
}

```

```

        catch (Exception e)
    {
        System.out.println("Inside catch");
    }
    finally
    {
        System.out.println("Inside finally");
    }
    System.out.println("After finally");
}

```

OP:-

Before try  
 inside try, before exception  
 inside catch  
 inside finally  
 After finally.

In java applications, we are able to write try-catch-finally inside try block, inside catch block and inside finally block.

Ex:-

```

try
{
    try
    {
        ...
    }
    catch (Exception e)
    {
        ...
    }
}
```

```

}
}
finally
{
}
}

catch (Exception e)
{
}
finally { } on
```

Ex:-②

```
try
{ }
```

```
Catch (Exception e)
{ }
```

```
try
{ }
```

```
Catch (Exception e)
{ }
```

```
finally
{ }
```

```
{ }
```

```
} finally
{ }
```

Static Keyword:-

The main purpose of static keyword is to improve sharability in java applications.

In java applications, there are three ways to use static keyword.

- ① Static Variables
- ② Static Methods
- ③ Static Blocks.

Ex:-③

```
try
{ }
```

```
Catch (Exception e)
{ }
```

```
finally
{ }
```

```
try
{ }
```

```
Catch (Exception e)
{ }
```

```
{ }
```

```
finally
{ }
```

```
{ }
```

```
} }
```

## ① Static Variables:-

It is recognized and initialized at the time of loading class bytecode to the memory.

Static variables must be declared as class level variables,  
Static variables must not be declared as local variables.

We can access static variables either by using class name directly or by using reference variable.

Static variables data will be stored in "Method Area".

In java apps, we can access current class static variables by using "this" keyword.

Ex:-  
Class A  
{  
 static int i=10;  
 void m1()  
 {  
 //static int j=20; → Error.  
 System.out("m1-A");  
 System.out("this.i");  
 }  
}  
Class Test  
{  
 public static void main(String[] args)  
 {  
 System.out(A.i);  
 A a=new A();  
 System.out(a.e);  
 a.m1();  
 }  
}

03/09/2015

(14)

### Static method:-

It is a normal java method, it will be recognized and executed the moment when we access that method.

→ Static methods are accessed either by using class name directly or by using reference variable.

→ Static methods are allowing only static members of the current class, static methods are not allowing non-static members of the current class.

→ Static methods are not allowing "this" keyword in their body, but to access current class static methods we are able to use "this" keyword.

Class A

Static int i=10;

int j=20;

Static void m1()

S.O.P("m1-A");

S.O.P(i);

// S.O.P(j); → Error

// S.O.P(this.i); → Error.

Void m2()

S.O.P("m2-A");

this.m1();

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```

class Test
{
    psvm (String []args)
    {
        A a = new AC();
        a.m1 ();
        a.m2 ();
        a.m1 ();
    }
}

```

### Static Block:-

It is a set of instructions, it will be recognized and executed at the time of loading the respective class bytecode.

- Static Blocks are able to allow only static members of the current class directly, static blocks are not allowing non-static members of the current class.
- "this" keyword is not allowed in static blocks.

### Syntax:-

```

static
{
    --- instructions ---
}

```

```

ex:- class A
{
    int i=10;
    static int j=20;
    static
    {
        S.O.P1n ("SB-A");
        // S.O.P1n (i); → Error
        S.O.P1n (j);
        // S.O.P1n (this.j); → Error
    }
}

```

```

class Test
{
    psvm (String []args)
    {
        A a = new AC();
    }
}

```

## JDBC [Java Database Connectivity]

As part of enterprise applications, it is convention to manage the details of the organisations including employee data, customers data, services data, clients details, . . .

In enterprise application development to manage the above specified data we have to use storage areas.

There are two types of storage Areas:-

1) Temporary Storage Areas:-

These are able to store data temporarily.

Ex - Buffers, java. object

2) Permanent Storage Areas:-

These are able to store data permanently.

Eg: file system, dbms, Data Warehouses.

Q: What are the differences between file system and DBMS?

① File system is platform dependent, it is not suitable for the platform independent programming language like java.  
DBMS is platform independent, it is suitable for all types of programming languages including java.

② File sys is able to store less data.

DBMS is able to ~~provide more security for the data~~ <sup>store data</sup>.

③ File sys is able to provide less security.

DBMS is able to provide more security for the data.

④ File sys is able to increase data redundancy.

DBMS is able to reduce data redundancy.

⑤ File system is not providing Query language support in order to perform database operations.

DBMS is providing Query language support to simplify database operations.

Q. What are the diff. between DBMS and Datawarehouse?

① DBMS is able to store less data.

Datawarehouse is able to store more data.

② DBMS is having fast insertion mechanisms, but it is not having fast retrieval mechanisms.

Datawarehouse is having fast retrieval mechanisms in the form of Data mining tech.

Q. What are the differences between Database and Database Management System?

① Database is a memory element to store data.

DBMS is a S/W system, it can be used to manage the data by storing it on Database and by retrieving it from database.

② Database is a collection of isolated data as a single unit.

DBMS is a collection of interrelated data and a set of programs to access data.

⇒ There are three types of DBMS.

① Relational Database Management Systems (RDBMS) :-

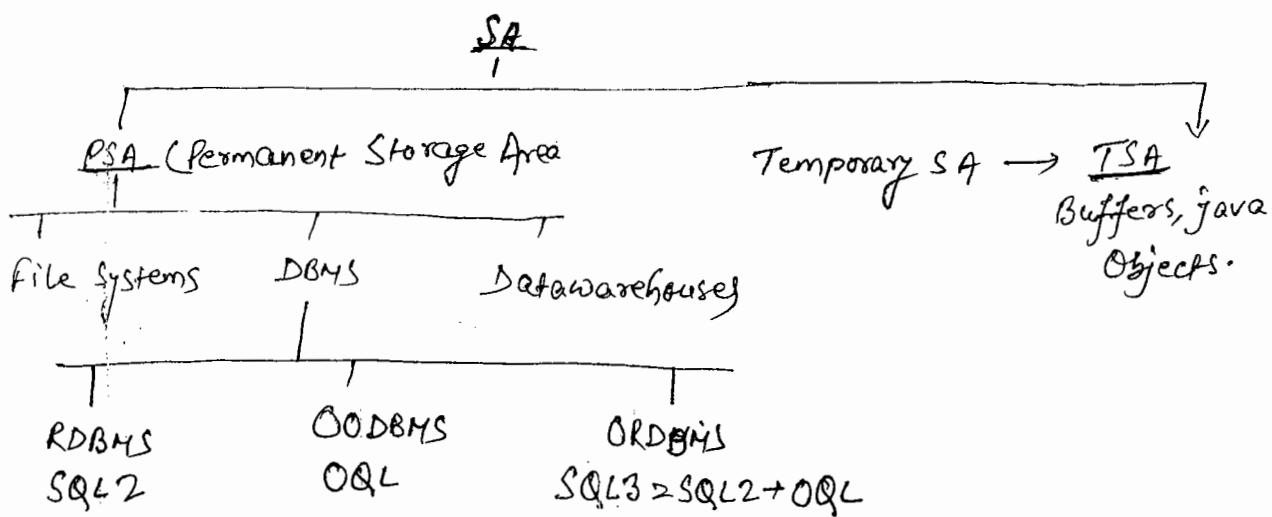
- It is able to represent the data in the form of tables.
- In RDBMS, we have to use SQL2 as Query Language to perform database Operations.

② Object Oriented Database Management System (OODBMS) :-

- It is able to represent data in the form of objects as per Object Oriented features.
- To perform database Operations in OODBMS we have to use OQL (Object Query Language) as query language.

③ Object-Relational Database Management System :-

- It is able to store data in the form of objects and tables, where to perform database Operations we have to use SQL3 as query language.
- where SQL3 is the combination of SQL2 & OQL.



05/09/2015

## Query Processing System :-

If we submit an SQL query to database then Database Engine will take that SQL query and Database Engine will execute that SQL query with the following steps:-

Steps:-

### ① Query Tokenization:-

This phase will take the provided SQL query as an input, it will divide the provided SQL query into no. of tokens and generates stream of tokens as an output.

### ② Query Parsing:-

The main intention of this phase is to check syntax errors in the provided SQL queries.

To check errors in SQL queries, Query parsing phase will construct a tree with stream of tokens called as "Query Tree". If query tree is success then no syntax errors are available in the provided SQL query. If query tree is not success then there are some syntax errors in the provided SQL query.

### ③ Query Optimization:-

This phase will take query as an input, it will perform no. of optimization mechanisms over query tree in order to reduce execution time and memory utilization, and generates optimized query tree.

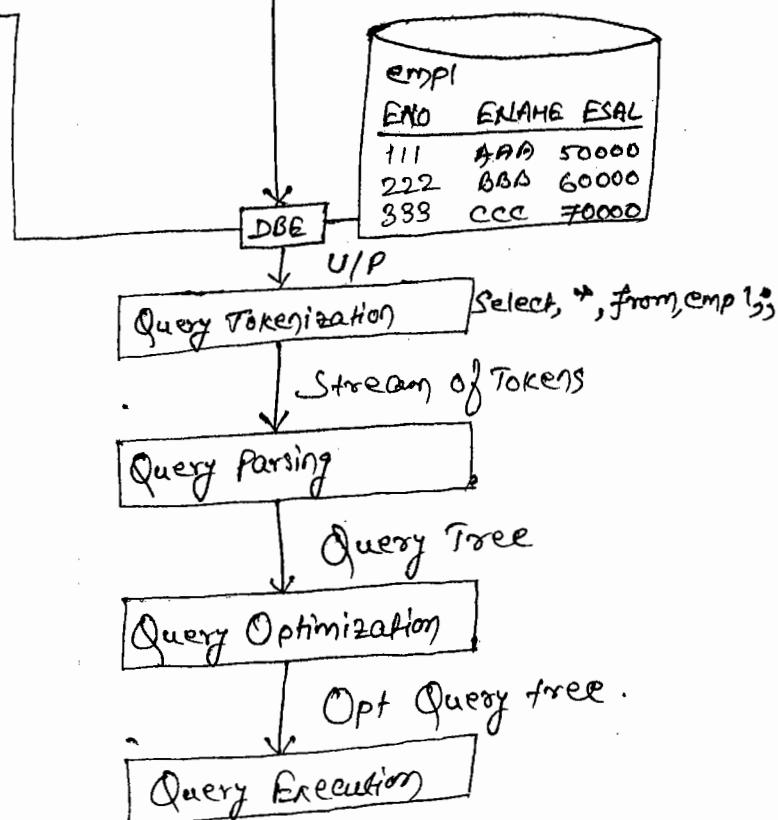
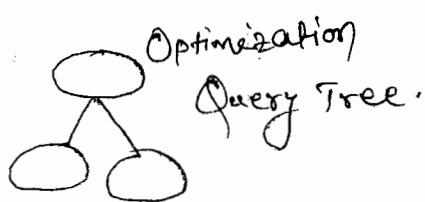
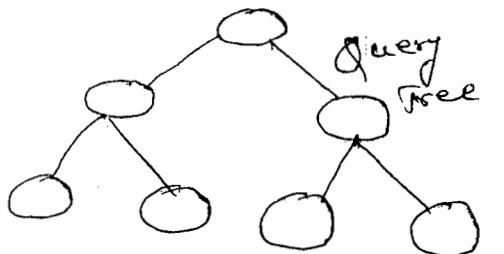
### ④ Query Execution:-

This phase will execute optimized query tree by having no. of interpreters, intentionally and performed the required database operation.

SQL: Select \* from emp;

ENO	ENAME	ESAL
111	AAA	50000
222	BBB	60000
333	CCC	70000

EMPLOYEE TABLE		
ENO	ENAME	ESAL
111	AAA	50000
222	BBB	60000
333	CCC	70000



## JDBC:-

- JDBC is a step by step process to interact with database from java applications in order to perform database operations.
- JDBC is a technology or an API or a collection of classes and interfaces, which will provide very good environment to connect with databases from java applications in order to perform database operations.
- JDBC was provided by JAVA on the basis of core libraries [JAVA SE].

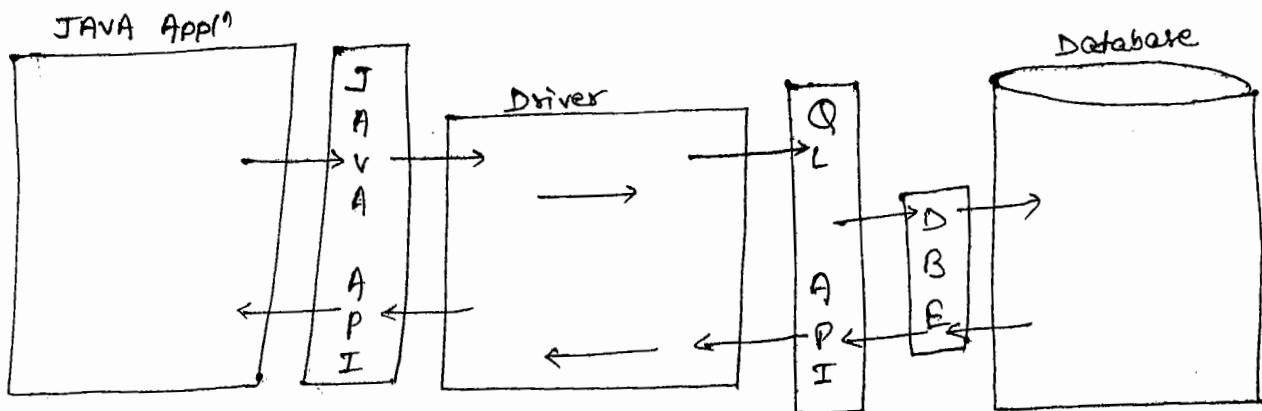
In general, in JDBC applications, we will provide database logic in java program as per JAVA API representations and we have to submit java represented database logic to database engine. In this context, Database Engine is unable to execute database logic, because, Database Engine is able to understand Query language API, not java API.

In the above context, to execute JDBC Applications we need a conversion mechanism to translate database logic from java representations to Database representations and from database representations to java representations, for this we have to use "Driver".

Driver is an interface existed b/w between java application and Database, it can be used to map JAVA representations to Database representations and Database representations to JAVA representations

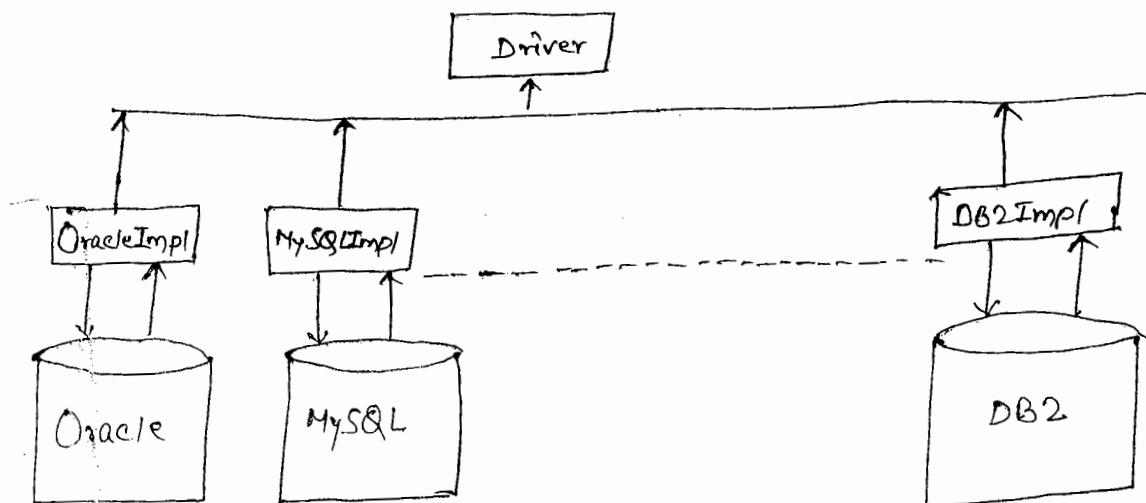
2

(A)



To provide driver as product, SUN Microsystems has provided java.sql.Driver as an interface and sun Microsystems has given an option to all the database vendors to provide their own implementation classes for java.sql.Driver interface.

With the above convention, all the database vendors have provided their own implementation classes for java.sql.Driver interface as part of their own database software.



In general, almost all the database vendors have provided their own drivers but all the drivers are divided into the following five types.

07-09-2015

T

Type-1 Driver

Type-2 Driver

Type-3 Driver

Type-4 Driver

Type-5 Driver

→ Where Type-5 driver is not fully implemented, it was provided by "Data Direct" company.

### Type-1 Drivers:-

Type-1 driver was provided by Sun Microsystems in the form of "Sun.jdbc.odbc.JdbcOdbcDriver" as a reference implementation for Driver interface.

SUN Microsystem has prepared Type-1 driver with the Interdependency of the Microsoft product ODBC Drivers.

ODBC driver was designed on the basis of the Microsoft provided native library, it will provide very good environment to interact with any types of database from "JdbcOdbc" Driver.

ODBC driver is very good compatible with Microsoft products, so, that Type-1 driver is very good compatible with Microsoft products, so, that Type-1 driver is less portable Driver.

<sup>3</sup>  
if we want to use Type-1 Driver in JDBC applications then we must install ODBC native library in our System.

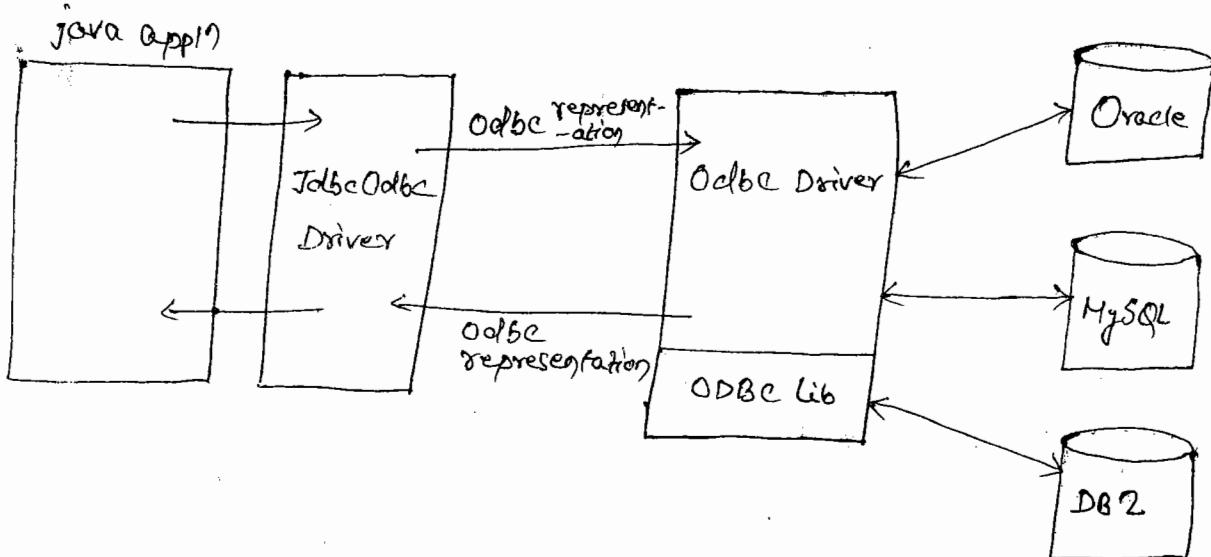
→ Type-1 driver is slower driver, because Type-1 driver must

require two times conversions [java to ODBC and ODBC to database representations] in order to interact with Database from java application.

Type-1 driver is suggestable for only simple JDBC application requirements, it is not suggestable for complex jdbc appn requirement, because it is model implementation for all the database vendors.

Type-1 Driver is suggestable for standalone applications, Type-1 Driver is not suggestable for web applications and distributed applications.

3



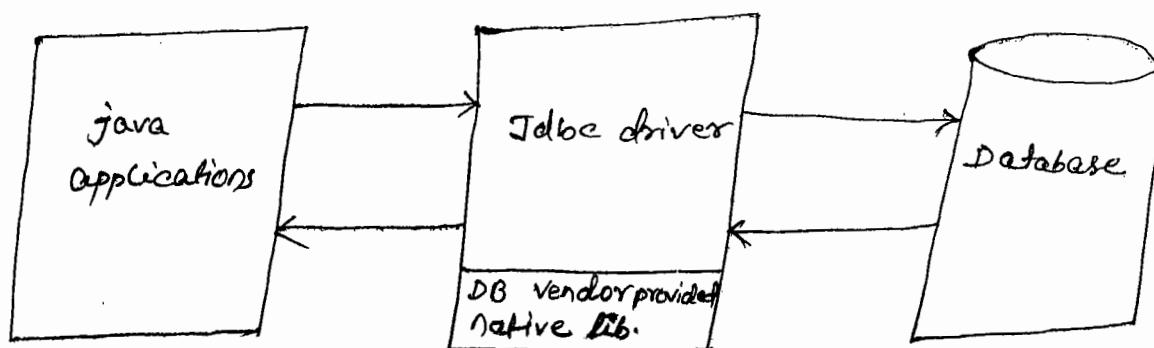
Note:- Type-1 driver is managed by Sun Microsystems upto JAVA 7 Version, Type-1 driver is removed from JAVA by Sun Microsystems from its JAVA 8 version.]

III

## Type-2 Driver:-

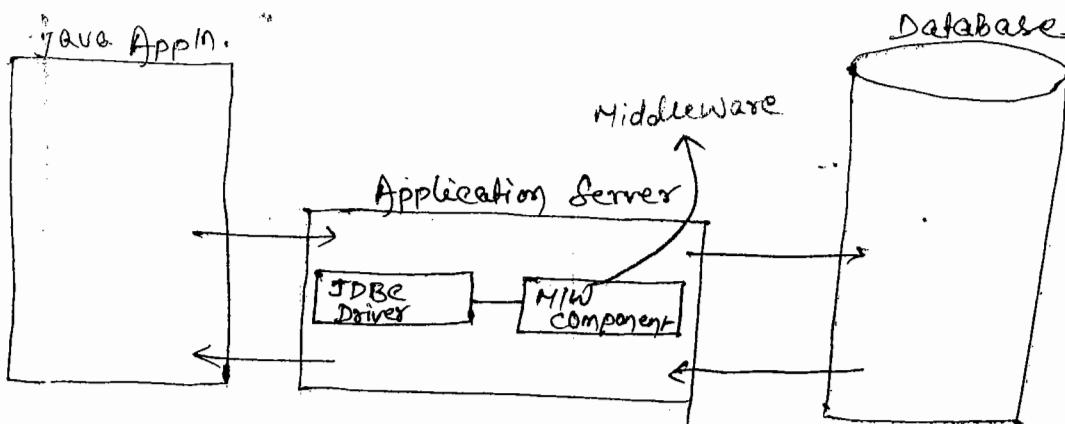
Type-2 driver is also called as "part Java/Part Native Driver" or simply "Native Driver".

- Type-2 driver was designed on the basis of java implementations and Database vendor provided native library.
- If we want to use Type-2 driver in JDBC applications then we have to install Database vendor provided native library.
- When compared with Type-1 driver, Type-2 driver is more portable driver, because, Type-2 driver is not required ODBC driver.
- When compared with Type-1 driver, Type-2 driver is faster driver, because, Type-2 driver does not require ~~two~~ two times conversions to connect with database from java applications.
- When compared with Type-1 driver, Type-2 driver is more <sup>(portable)</sup> performance driver.
- Like Type-1 driver, Type-2 driver is suggestable for standalone Applications, it is not suggestable for web applications and distributed Applications.
- Type-2 driver is costfull driver, it is not economical driver.



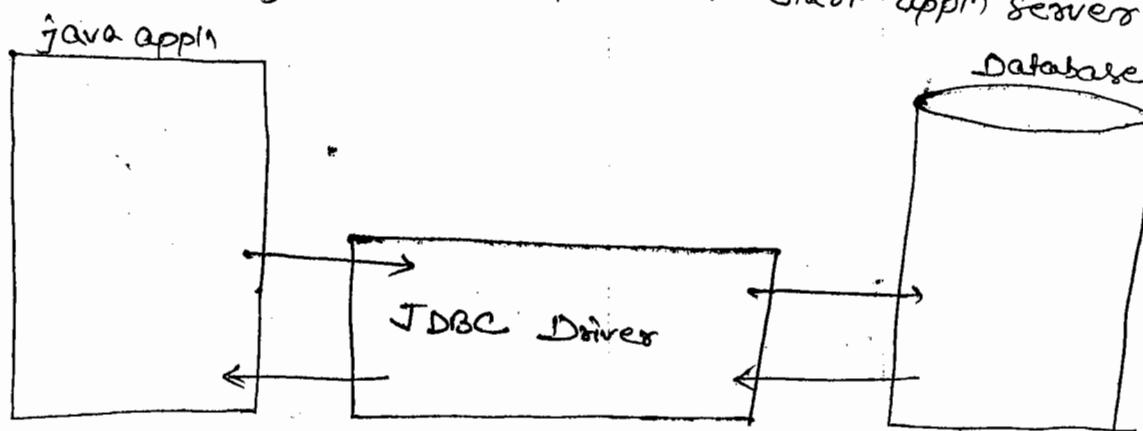
### Type-3 Driver:-

- Type-1 and Type-2 drivers are suggestable for only standalone applications, but Type-3 driver was prepared for web apps<sup>1</sup> and distributed apps.
- Type-3 driver is also called as "middleware database server access driver" or simply "Network Driver".
- Type-3 driver will provide very good environment to connect with multiple databases from multiple java apps.
- Type-3 driver is more portable driver when compared with Type-1 and Type-2 driver, because, Type-3 driver is not required ODBC native library and database vendor provided native library.
- Type-3 driver is faster driver when compared with Type-1 and Type-2 drivers, because Type-3 driver is not required two times conversions and not required to load database vendor provided native library.
- Type-3 driver will improve JDBC apps' performance when compared with Type-1 and Type-2 drivers.
- Type-3 driver is not suggestable for standalone applications.



### Type-4 Driver:-

- Type-1 and Type-2 driver are suggestables for only standalone applications, Type-3 driver is suggestable for only web application and distributed apps, but Type-4 driver is suggestable for both standalone apps, web apps, and distributed apps.
- Type-4 driver is also called as thin driver and pure java driver, because Type-4 driver was prepared completely on the basis of ~~java application~~ implementation.
- Type-4 driver is frequently used driver in appn development.
- Type-4 driver is economical driver, it is not costfull driver.
- Type-4 driver is more portable driver, because it is not required database native library, database vendor provided native library and application servers.
- Type-4 driver is faster then compared with Type1, Type-2 and Type-3 drivers, because, it should not required two time conversions, not required to load database vendor provided native library and not require to start appn server.



## First Steps to Design Jdbc Application:-

- ① Load and Register Driver
- ② Establish connection between java application and database.
- ③ Create Either Statement or PreparedStatement or CallableStatement Objects as per the application requirement.
- ④ Write and execute SQL Queries
- ⑤ Close the connection.

### ① Load and register Driver :-

- In jdbc applications, if we want to load and register driver, first we have to make available driver class to JVM.
- Driver is an interface provided by SUN Microsystems and its implementation classes are provided by database vendors in their respective database software.
  - In the above context, to make available driver class from Database software to Jdbc application, we have to set "Classpath" environment variable.
  - In Jdbc applications, if we want to use Type-1 driver then it is not required to set "Classpath" environment variable, because Type-1 driver class is provided by SUN Microsystems in the same java software.
  - In the Jdbc applications, if we want to load driver class then we have to use the following method from java.lang. - Class class.

```
public static Class forName(String driver-Class-name) throws
java.lang.ClassNotFoundException;
```

Ex:- `Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`

When JVM encounters the above instruction then JVM will perform the following actions.

- ① JVM will take the provided driver class name from Class.forName(-) method.
  - ② JVM will search for driver class .class file at current location, at java predefined library and at the locations referred by "Classpath" environment Variable.
  - ③ If the required driver .class file is not available at all the above specified locations then JVM will rise an exception like "java.lang.ClassNotFoundException".
  - ④ If the required driver .class file is available at either of the above locations then JVM will load driver class bytecode to the memory.
  - ⑤ At the time of loading driver class bytecode to the memory, JVM will execute a static block where inside static block JVM will execute:  

"DriverManager.registerDriver(--)" method.
- With the above method execution Only, Java Driver will be registered to our JDBC Application.
- ⑥ After loading driver class bytecode to the memory, JVM will get metadata of the loaded driver and JVM will store metadata by creating an Object for java.lang.Class class in heap memory.

If we want to use Type-1 driver in JDBC applications then we have to configure Microsoft provided ODBC Driver in our system, for this we have to use the following steps:

- ① Click on "Start" button or Search.
- ② Select "Control panel".
- ③ Select System and Security.
- ④ Select "Administrative tool".
- ⑤ Select "ODBC Data Sources (32-bit)" and double click.
- ⑥ Select "User DSN" and Click on "Add" button.
- ⑦ Select "Microsoft ODBC for Oracle".
- ⑧ Click on "Finish" button.
- ⑨ Provided "Data Source Name" [AMARENDRA]
- ⑩ Click on "OK" button.

(23)

Static Variable data,  
driver class bytecode → Method Area  
Local variable data → Stack Memory  
instance variable objects created →  
Heap memory.

Note:- The above DSN name creation for Oracle is possible in  
Only windows 32 bit Operating System, it is not possible in windows  
64-bit Operating System.]

Note:- If we use JAVA6 version i.e. JDBC 3.0 version then Load and  
register Type-1 driver Step is Optional in jdbc application → Imp

Note:- If we use JAVA8 version i.e. JDBC 3.0 version and Oracle  
11g/11xe version then Load and Register Type-4 driver Step is  
Optional. → Imp

In the above two cases, if we are not performing Load  
and register driver step then JVM will load the respective  
driver class bytecode to the memory just before establishing the  
connection between java application and database.

To design jdbc application, java has provided the following  
predefined library in the form of java.sql package.

- Driver [Interface]
- DriverManager [Interface]
- Connection [Interface]
- Statement [Interface]

10-09-2015

- PreparedStatement [Interface]
- CallableStatement [Interface]
- ResultSet [Interface]
- ResultSetMetaData [Interface]
- DatabaseMetaData [Interface].

## 2) Establish Connection between Java application and Database:-

In JDBC applications, if we want to establish the connection between java applications and databases then we have to use the following method from java.sql.DriverManager Class.

```
public static Connection getConnection(String driverURL, String  
dbUserName, String dbPwd) throws java.sql.SQLException
```

→ Where driver URL is changed from driver to drivers, but its format is fixed.

Main-Protocol-Name : Sub-Protocol-Name ? DB-Name.

→ Where Main-Protocol-Name is fixed by all the drivers that is "JDBC".

Ex:-

Jdbc:odbc:dsn-Name (Type-1 Driver)

Jdbc:Oracle:thin:@localhost:1521:xe (Type-4 from Oracle)

Jdbc:mysql://localhost:3306/db-name (Type-4 from MySQL)

Ex:-

```
Connection Con = DriverManager.getConnection("jdbc:odbc:dsn-Name",  
"System", "alurga");
```

P.T.O.

→ When jvm encounters the above instruction, jvm will execute getConnection() method, where getConnection() method will execute connect() method, where connect() method will establish socket connection between java applications and application database as per the driver URL which we provided with this. getConnection() method will return Connection Object or object to the Connection reference value.

Q. In java application, it's not possible to create objects for interface but how it's possible for getConnection() method to create object for Connection object % interface?

→ In java appn, Connection is an interface provided by Sun Microsystems but its implementation classes are provided by all the Database Vendors.

→ In jdbm appn getConnection() method is not creating object directly for connection interface, getConnection() method is creating object for the implementation class of connection interface provided the respective database vendor.

Ex:-

public interface Connection { // provided by sun microsystem }

----- Some abstract methods declaration -----

public class Oracle\_Connection\_Impl implements Connection {  
// provided by Oracle }

----- Implementation for all the abstract methods -----

}

11-09-2015

```
public class DriverManager {  
    public static Connection getConnection(---, ---, ---) {  
        --- Connect();  
    }
```

```
    Connection con = new Oracle_Connection_Impl();
```

```
    return con;  
}
```

```
}
```

```
Class JobApp {
```

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

```
        Connection con = DriverManager.getConnection(---, ---, ---)
```

```
}
```

```
}
```

```
==
```

③ Create Statement Object or PreparedStatement Object or CallableStatement Object as per the requirement :-

→ In job app after loading the driver and after establishing the connection between java application and database we have to write SQL queries and we have to execute SQL queries. To write and execute SQL queries in job applications, we need some predefined library, here the required predefined library is provided by JDBC API in the form of Statement, PreparedStatement and CallableStatement.



Q: What is the difference between Prepared Statement, Statement and Callable Statement?

- In jobe applications, when we have a requirement to execute all the SQL queries independently, then we have to use Statement.
- In jobe applications, when we have a requirement to execute the same SQL queries in the next sequence, where to improve jobe application performance we have to use "PreparedStatement".
- In jobe applications, when we have a requirement to access stored procedures and functions available at database from java applications then we have to use CallableStatement.

3

To create Statement Object in jobe Applications we have to use the following method from java.sql.Connection.

public Statement createStatement() throws SQLException

Ex:-

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

Note:- In jobe applications, createStatement() method is not creating object for Statement interface directly, createStatement() method is creating object for implementation class of Statement interface and return the generated reference value.

~~3~~

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

#### ④ Write and execute SQL Queries:-

→ In JDBC applications, if we want to write and execute SQL queries, we have to use the following methods from java.sql.Statement.

- ① executeQuery(--)
- ② executeUpdate(--)
- ③ execute(--)

Q. What are the diff. b/w executeQuery() method, executeUpdate() method and execute() method?

In JDBC applications, executeQuery() method, can be used to execute select SQL query in order to fetch the data from database table.

public ResultSet executeQuery(String SQL\_query) throws SQLException;

Ex:-

```
ResultSet rs = st.executeQuery("select * from emp");
```

When JVM encounter the above instruction, JVM will take the provided select SQL query from executeQuery() method, ~~JDBC driver will~~ → JVM will send that SQL query to JDBC Driver, where JDBC driver will submit select SQL query to Connection, where Connection will carry that select SQL query to Database engine.

At Database, database engine will execute select SQL query, retrieves data from database tables and send that retrieved data to connection, where connection will carry that result to JDBC driver, where JDBC driver will send that retrieved

data back to java application in the form of ResultSet object.

In jdb applications, executeUpdate() method can be used to execute non-select SQL queries like create, insert, update, delete, drop, ... in order to perform the respective database operations.

`public int executeUpdate(String SQL-Query) throws SQLException.`

Ex:-

```
int rowCount = st.executeUpdate("Update emp1 set esal=esal+500 where esal < 10000");
```

→ When JVM encounter the above instruction, JVM will send the specified non-select SQL query to database Engine through JDBC Driver and connection, where Database Engine will execute non-select SQL query, performs database operation and identify the no. of records which are effected that RowCount value and sends RowCount value back to java application.

3

In jdb applications, execute() method can be used to execute both select and non-select SQL queries.

`public boolean execute(String SQL-Query) throws SQLException.`

Ex:-

```
boolean b1 = st.execute("Select * from emp1");
S.O.P(b1); // O/P: true.
```

```
boolean b2 = st.execute("Update emp1 set esal=esal+500 where
esal < 10000");
```

S.O.Pn (52); // Op:- false.

When JDB encounter execute() method with select SQL query then JDB will send that select SQL query to Database Engine, where Database Engine will execute that SQL query and send the retrieved data to java app in the form of ResultSet object, but, execute() method will return "true" value as per its predefined implementation part.

When JDB encounter execute() method with non-select SQL query JDB will send that non-select SQL query to Database Engine, where Database Engine will manipulate no. of records and return rowCount value to java application but, execute() method will return "false" value.

#### ⑤ Close the Connection:-

In JDBC applications, at the end of the programs it is convention to close Statement and Connection, for this we have to use the following methods.

public void close() throws <sup>SQL</sup>Exception

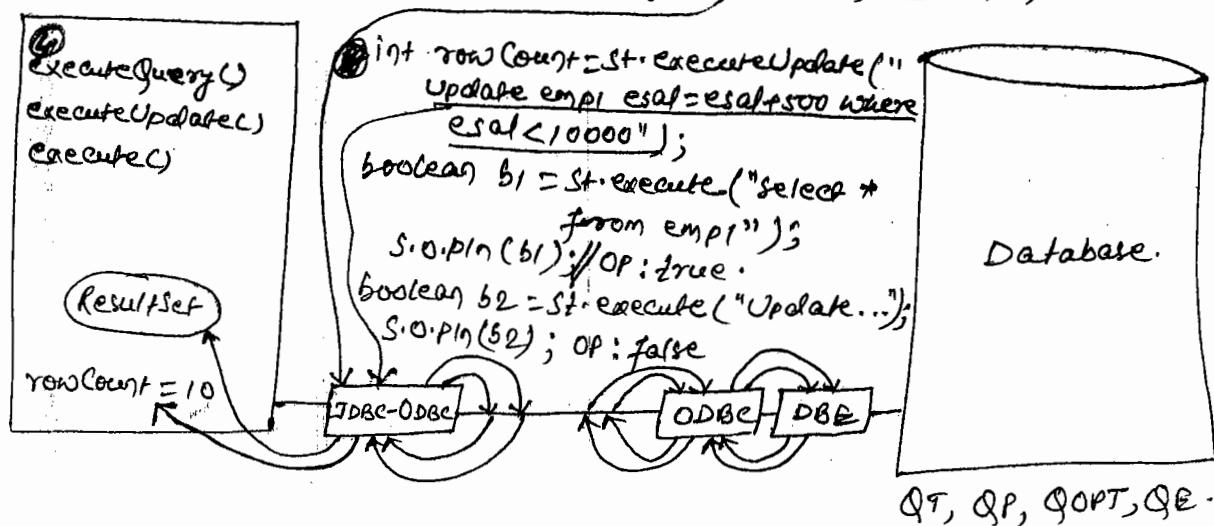
Ex:- St.close();

Con.close();

Z

① Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");

② ResultSet rs = st.executeQuery ("Select \* from emp");



② Connection con = DriverManager.getConnection ("jdbc:odbc:naq", "system", "elurga");

③ Statement st = con.createStatement();

④ st.close();

con.close();

≡

### SRI RAGHAVENDRA XEROX

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

14-09-2015

Ex:- ①

```
// Import Statements
import java.sql.*;
import java.io.*;
class JdbcApp1
{
    public void main(String [ ] args) throws Exception
    {
        // Load & Register Driver
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        // Establish connection between java application and DB.
        Connection con = DriverManager.getConnection("jdbc:odbc:Apnar",  

            "System", "durga");  

        // Create Statement Object
        Statement st = con.createStatement();
        // Take Table name as dynamic input
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Table name : ");
        String fname = br.readLine();
        // Write SQL Query.
        String sqlQuery = "Create table " + fname + "(eno number primary  

            key,ename varchar2(10),esal number, eaddr varchar2(10))";
        // Execute SQL Query.
        st.executeUpdate(sqlQuery);
        System.out.println(fname + " table created successfully");
        // Close Statement and Connections.
        st.close();
        con.close();
    }
}
```

Note:- To run the above program we must provide "Amarendra" as DSN Name in ODBC Driver Configuration.

We must execute the above program in Only JAVA7 version, not in JAVA8 version, because Sun.jdbc.ODBC.JdbcODBCDriver has removed in JAVA8 version.] .

If we want to use Type-4 driver provided by Oracle then we have to use the following driver class name and driver URL.

Driver-Class: oracle.jdbc.OracleDriver[Oracle 9/10/11]  
oracle.jdbc.driver.OracleDriver[Oracle 8i].

Driver-URL: jdbc:oracle:thin:@localhost:1521:xe[Oracle10xe/11xe].  
jdbc:oracle:thin:@localhost:1521:ORCL[Oracle10g/11g].

To use the above driver class <sup>name</sup> in JDBC applications we have to set "CLASSPATH" environment variable to the respective java file which contains the driver class.

Oracle11g/11xe → ojdbc6.jar[JAVA6 and above]  
→ ojdbc5.jar[JDK5+0]

Oracle10g/10xe → ojdbc4.jar

Oracle8i/9i → classes12.jar, classes11i.jar, classes14.jar

Ex:-

Set CLASSPATH = C:\oracle\app\oracle\product\11.2.0\server\jdbc\lib  
ojdbc6.jar

Ex:- 2

```
import java.io.*;
import java.sql.*;
class JdbcApp2
{
    public static void main(String[] args) throws Exception
    {
```

```
Class.forName("Oracle.jdbc.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:",
@localhost:1521:xe", "system", "deurga");
Statement st=con.createStatement();
BufferedReader br=new BufferedReader(new InputStreamReader
(System.in));
while(true)
{
    System.out.print("Employee Number :");
    int eno=Integer.parseInt(br.readLine());
    System.out.print("Employee Name :");
    String ename=br.readLine();
    System.out.print("Employee Salary :");
    float esal=Float.parseFloat(br.readLine());
    System.out.print("Employee Address :");
    String eaddr=br.readLine();
    st.executeUpdate("insert into emp values ('"+eno+"','"+ename",
"+"+esal+"','"+eaddr+"')");
    System.out.println("Employee Inserted successfully");
    System.out.println("One more Employee (yes/no)?");
    String option=br.readLine();
    if(option.equals("no"))
    {
        break;
    }
}
st.close();
con.close();
```

Ex. 3

```

import java.io.*;
import java.sql.*;
public class JdbcApp3
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.oracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "durga");
        Statement st = con.createStatement();
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter Bonus Amount :");
        int bonusAmt = Integer.parseInt(br.readLine());
        System.out.println("Enter Salary Range :");
        float salRange = Float.parseFloat(br.readLine());
        int rowCount = st.executeUpdate("Update emp1 set esal=esal+" +
                                         "+bonusAmt" + " where esal<" + salRange);
        System.out.println("No. of Records Updated : " + rowCount);
        st.close();
        con.close();
    }
}

```

Ex-  
4

```
import java.io.*;
import java.sql.*;
public class JDBCApp4{
    public static void main(String[] args) throws Exception{
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "deurga");
        Statement st = con.createStatement();
        BufferedReader br = new BufferedReader(new InputStreamReader(
            (System.in)));
        System.out.println("Salary Range: ");
        float salRange = Float.parseFloat(br.readLine());
        int rowCount = st.executeUpdate("delete emp where esal <=" +
            salRange);
        System.out.println("Records Deleted :" + rowCount);
        st.close();
        con.close();
    }
}
```

Q. In jobe applications, if we execute DML SQL queries like insert, update and delete by using executeUpdate() then database engine will manipulate some records and Database Engine will sent a particular rowCount value to java appn, where at java appn executeUpdate() method will return the generated rowCount value. If we execute DDL SQL queries like create alter and drop by using

`executeUpdate()` method then database engine will not perform manipulations on no. of records and the Database Engine will not send any rowCount value. In this context, which integer value would be returned by `executeUpdate()` method in Jobe Applications?

- In Jobe Applications, we can use `executeUpdate()` method to execute both DML and DDL SQL queries and it will return an integer value representing the no. of records which are manipulated at database table.
- If we execute DML SQL queries like insert, update and delete. by using `executeUpdate()` method then database engine will perform manipulations on database table and generate rowCount value to java appn, where at java application `executeUpdate()` method return the generated rowCount value.
- If we execute DDL SQL queries like create, alter and drop by using `executeUpdate()` method the database engine will not perform manipulations on database tables, it will not any rowCount value to java application. In this Context, return value of `executeUpdate()` method is completely depending on the type of drive which we used.
- In the context, if we use Type-1 driver provided by SUN Microsystems then `executeUpdate()` Method will return '-1' value.
- In the above context, if we use Type-4 driver provided by Oracle then `executeUpdate()` method will return '0' value.

Ex:- 5 → Type-1 Driver.

```
import java.sql.*;
public class JdbcApps {
    public static void main (String[] args) throws Exception {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection ("jdbc:odbc:Amey",
                                                    "System", "durga");
        Statement st = con.createStatement ();
        int rowCount1 = st.executeUpdate ("Create table emp10
                                         (eno number)");
        int rowCount2 = st.executeUpdate ("drop table emp10");
        System.out.println (rowCount1);
        System.out.println (rowCount2);
        st.close();
        con.close();
    }
}
```

Ex:- 6 import java.sql.\*;
public class JdbcApps {
 public static void main (String[] args) throws Exception {
 Class.forName ("oracle.jdbc.oracleDriver");
 Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:xe",
 "System", "durga");
 Statement st = con.createStatement ();
 int rowCount1 = st.executeUpdate ("Create table emp10 (eno number)");
 int rowCount2 = st.executeUpdate ("drop table emp10");
 System.out.println (rowCount1);
 System.out.println (rowCount2);
 st.close();
 con.close();
 }
}

### ResultSet:-

In JDBC applications, if we execute select SQL query by using `executeQuery()` method then JVM will send that select SQL query to Database Engine, Where Database Engine will execute select SQL query, Database Engine will send the fetched data to java app. At java application fetched data will be stored in the form of `ResultSet` object.

When `ResultSet` object is created automatically a cursor will be created before the first record called as "ResultSet" Cursor".

→ In JDBC applications, if we want to retrieve data from `ResultSet` Object then we have to use the following steps.

- ① Check whether next record is available or not from the current cursor position. If next record is available then move cursor to next record position.  
To perform this action, we have to use the following method from `java.sql.ResultSet`.

`public boolean next() throws SQLException;`

- If next record is available then `next()` method will move cursor to next record position and return "true" value.
- If next record is not available then `next()` method will return "false" value.

- ② After getting cursor to next record position then read data from the columns individually.

To read a particular column value, we have to use the following methods.

public xxx getxxx (int columnIndex) throws SQLException  
 public xxx getxxx (String columnName) throws SQLException.  
 Where xxx may be byte, short, int, ...

The above process must be repeated upto all the records which are available in Resultset Object in order to get data from Resultset Object.

ResultSet rs = st.executeQuery ("Select \* from emp");

	int	String	float	String
	1	2	3	4
before record				
after record				
after 1st record				
after 1st record and before 2nd record				
cursor				
	ENO	ENAME	ESAL	EADDR
	111	AAA	5000	Hyd
	222	BBB	6000	Hyd
	333	CCC	7000	Hyd

while (rs.next()) {

```

    System.out.println(rs.getInt(1));
    System.out.println(rs.getString(2));
    System.out.println(rs.getFloat(3));
    System.out.println(rs.getString(4));
  }
  =====

```

Ex:- 7

```

import java.sql.*;
public class JDBCApp7 {
  public void main (String [] args) throws Exception {
  }
}

```

```

    // Class.forName("Oracle.jdbc.OracleDriver");
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:",
                                                "@localhost:1521:xe", "system", "kings");
    Statement st =
    ResultSet rs = st.executeQuery("Select * from emp");
    System.out.println("ENO"+ENAME+ESAL+DEADDR");
    System.out.println("-----");
    while(rs.next()){
        System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getFloat(3)+" "+rs.getString(4));
    }
    st.close();
    con.close();
}

```

Q In jdbc applications, we will use executeUpdate() method to execute non select SQL queries, but if we execute select SQL query through executeUpdate() method then what would be the result in jdbc applications?

→ If we execute select SQL query by using executeUpdate() method then JVM will send the provided select SQL query to Database Engine, where Database Engine will fetch data from the table and return to java application in the form of ResultSet object, but executeUpdate() is expecting rowCount value from Database Engine.

In the above situation, rising an exception or not is completely depending on the type of driver which we used.

→ for the above requirement, if we use Type-1 driver then JVN will rise an exception like

"java.sql.SQLException: Now rowCount was produced."

for the above requirement if we use Type-4 driver provided by Oracle then JVN will not rise any exception, where JVN will return the no. of records which are available in the generated ResultSet object as rowCount value.

(Note:-) In the above situation, if we want to get the generated ResultSet Object reference explicitly then we have to use the following method from java.sql.Statement.

public ResultSet getResultSet() throws SQLException;

Eg:- (Type-1 Driver) → (javat version) → it's & version class not found + exception we will found.

```
import java.sql.*;
public class jdbcapp {
    public static void main (String [] args) {
        Statement st = null;
        try {
            Class.forName("sun.jdbc.odbc.jdbcOdbcDriver");
            Connection con = DriverManager.getConnection ("jdbc:odbc:nag",
                "System", "durga");
            st = con.createStatement();
            int rowCount = st.executeUpdate ("Select * from emp");
        }
    }
}
```

*to make available statement for try block as well as catch block put null value is mandatory because local variable don't have garbage value*

```

    catch (Exception e)
    {
        e.printStackTrace();
    }
    try
    {
        ResultSet rs = st.executeQuery();
        System.out.println("ENO" + ENAME + "ESAL" + EADDR);
        System.out.println("-----");
        while (rs.next())
        {
            System.out.println(rs.getInt("ENO") + " " + rs.getString("ENAME") + " " + rs.getFloat("ESAL") + " " + rs.getString("EADDR"));
        }
    }
    catch (Exception e2)
    {
        e2.printStackTrace();
    }
}

```

we are using try catch because if any exception occurred inside the try then we are able to catch the flow of execution.

here, again we are using the try-catch inside catch because there may a chance to occur any exception.

Ex:- 9

```

import java.sql.*;
public class JdbcApp9
{
    public void main(String[] args) throws Exception
    {
        Class.forName("Oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "Durga");
        Statement st = con.createStatement();
        int rowCount = st.executeUpdate("Select * from emp");
    }
}

```

```

S.O.Pn("Row Count :" + rowCount);
ResultSet rs=st.getResultSet();
S.O.Pn("ENO1 FNAME1 TNAME1 BSAL1 FADDR17");
S.O.Pn("-----");
while(rs.next()){
    S.O.Pn(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getFloat(3) +
           " " + rs.getString(4));
}
st.close();
con.close();
}
}

```

Q. In jobe applications, we will use executeQuery() method to execute Select SQL queries, but, if we execute non-select SQL queries with executeQuery() method then what would the result in jobe applications?

⇒ In jobe Apps, if we pass non-select SQL query as parameter to executeQuery() method then job will send the specified non-select SQL query to Database Engine, where Database Engine will perform manipulations on no. of records and return rowCount but executeQuery() method is expecting ResultSet Object.

→ In the above situation, rising an exception or not rising an exception is completely depending on the type of driver which we used.

→ for the above requirement, if we use Type-1 driver then JVM will rise an exception like

"java.sql.SQLException: No ResultSet was produced".

→ for the above requirement, if we use Type-4 driver provided by Oracle then JVM will not rise any exception, where JVM will prepare the default ResultSet Object.

Note:- In the above situation, to get the generated rowCount value we have to use the following method from java.sql.Statement.

public int getUpdateCount() throws SQLException.

$\Sigma$

### Ex:- (Type-2 Driver) :-

```

import java.sql.*;
public class JdbcApp10 {
    public static void main(String[] args) {
        Statement St = null;
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con = DriverManager.getConnection("jdbc:odbc:nag", "system",
                "durga");
            St = con.createStatement();
            ResultSet rs = St.executeQuery("Update emp1 Set esal=esal+500
                Where esal<10000");
        }
    }
}

```

```

    catch (Exception e) {
        e.printStackTrace();
    }
    try {
        int rowCount = st.executeUpdate();
        System.out.println("Row Count :" + rowCount);
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}
}

```

Ex:-

20/09/2015

```

import java.sql.*;
public class JdbcApp11 {
    public static void main (String [] args) throws Exception {
        Class.forName("Oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:
            @localhost:1521:xe", "System", "durga");
        Statement st = con.createStatement ();
        ResultSet rs = st.executeQuery ("Update emp1 set esal=esal-
            500 where esal < 10000");
        int rowCount = st.executeUpdate ();
        System.out.println("Row Count :" + rowCount);
        st.close ();
        con.close ();
    }
}

```

In JDBC applications, we will use `execute()` method to execute both select and non-select SQL queries.

If we pass select SQL query as parameter to `execute()` method then JVM will send that select SQL query to database engine, where database engine will execute select SQL query, where database engine will fetch the data from database table and return to Java application in the form of `ResultSet`. Set object, but, as per the internal implementation of `execute()` method JVM will return "true" value.

Ex:-

```

import java.sql.*;
public class JdbcApp12 {
    public static void main(String[] args) throws SQLException {
        Class.forName("Oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@"
                + "localhost:1521:xe", "System", "durga");
        Statement st = con.createStatement();
        boolean b = st.execute("Select * from emp");
        System.out.println(b);
        ResultSet rs = st.getResultSet();
        System.out.println("ENO\ZENAME\TESAL\TEADDR");
        System.out.println("-----");
        while (rs.next()) {
            System.out.println(rs.getInt(1) + "\t" + rs.getString(2) + "\t" + rs.getFloat(3) +
                "\t" + rs.getString(4));
        }
    }
}

```

```

    St.close();
    Con.close();
}
}
if (rs.next())
}
}

```

If we pass non-select SQL query as parameter to execute() method then it will send that non-select SQL query to Database Engine, where Database Engine will perform manipulation on database table and returnRowCount Value to java appM, but, execute() method will return "false" value. In this Context, if we want to get rowCount Value explicitly then we have to use getUpdateCount() method.

Ex:-

```

import java.sql.*;
public class jobbeApp1 {
    public static void main(String[] args) throws Exception {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection Con = DriverManager.getConnection("jobbe:oracle:thin:@localhost:1521:xe", "System", "clurga");
        Statement St = Con.createStatement();
        boolean b = St.execute("Update emp1 set esal=esal-500
                               Where esal<10000");
        System.out.println("Row Count :" + St.getUpdateCount());
        St.close();
        Con.close();
    }
}

```

21/09/2018 (26)

## Eclipse:-

- ① Home: IBM Canada [Initially]  
Eclipse Foundations [2003]
- ② Website: www.eclipse.org
- ③ Objective: To Simplify JAVA/J2EE Application Development.
- ④ Initial Version: Eclipse 3.0 [2004]
- ⑤ Other Versions: Eclipse 3.5, 3.6, 3.7  
[Helios, Cheetah, Indigo]
- ⑥ Latest Version: Eclipse 4.2 [1st March, 2013] [Juno]
- Eclipse 4.3.1 [26 June, 2013] [Kepler]
- Eclipse 4.4 [9th Aug, 2013] [Luna]
- Eclipse 4.5 [June 28, 2015] [Mars]
- Eclipse 4.6 [June 23, 2016] [Neon]
- ⑦ Designed On: Java platform
- ⑧ Type: Open Source Software

## Installation Process:-

- ① Download eclipse-jee-Mars-SR-win32.zip
- ② Extract eclipse-jee-Mars-SR-win32.zip file at the required location
- ③ After Step 2, we will find eclipse folder i.e. installation folder.
- ④ Double click on eclipse icon available at "C:\eclipse" location.
- ⑤ After Step 4, we will get a window to select workspace, where specify the location where we want to manage all the applications.

2

## Steps to prepare JDBC Applications:-

### ① Create new java project:

- ① Select "File" in menu
- ② Select "New"
- ③ Select "Project"
- ④ Click on "Next" button
- ⑤ Provide project Name: JdbcApp and Click on "Next" button.
- ⑥ Click on "Finish" button
- ⑦ Click on "Yes" button.

### ② Add Driver jar file to project:-

- ① Right Click on project
- ② Select "Properties" [Last Option]
- ③ Select "java Build Path".
- ④ Select "Libraries".
- ⑤ Click on "Add External Jars".
- ⑥ Select "ojdbc6.jar" file and click on "Ok" button.
- ⑦ Click on "Ok" button.

### ③ Create java Class with main() method:-

- ① Right click on "src".
- ② Select "New".
- ③ Select "Class".
- ④ Provide the following

Package Name: com.durgasoft

Class Name: JdbcApp

Select "public static void main(String[] args)"

- ④ Write a JDBC program in the generated main class:-

— Write our own JDBC program —

- ⑤ Run Jdbc project:-

- ① Right click on "project".
- ② Select "Run as".
- ③ Select "java Application".

Ex 14 :-

```

import java.io.FileOutputStream;
import java.sql.*;
public class JdbcApp14 {
    public void main(String[] args) throws Exception {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@"
            + "localhost:1521:xe", "system", "durga");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("Select * from emp");
        String[] header = {ENO, ENAME, ESAL, EADDR};
        String data = "<html><body><center><br><br>";
        data = data + "<table border='1'><tr>";
        for (int i = 0; i < header.length; i++) {
    
```

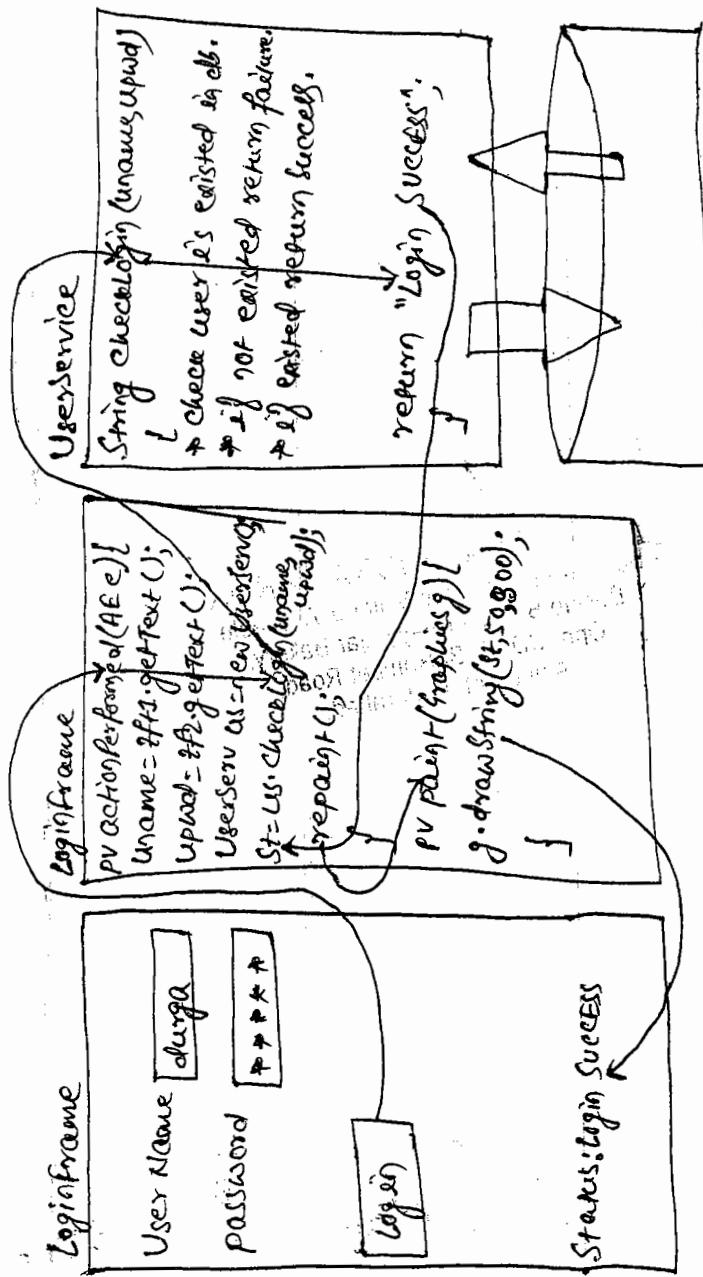
SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Amerpet, Hyderabad.

```
data = data + "<td><font color='red' size='5'><b>" +  
data = data + header[i] +  
data = data + "</b></font></td>";  
}  
data = data + "</tr>";  
while(rs.next()) {  
    data = data + "<tr>";  
    for(int i=1; i<=header.length; i++) {  
        data = data + "<td>";  
        data = data + rs.getString(i);  
        data = data + "</td>";  
    }  
    data = data + "</tr>";  
}  
data = data + "</table></center></body></html>";  
FileOutputStream fos = new FileOutputStream("C:/tabe/emp.html");  
byte[] b = data.getBytes();  
System.out.println("Data is transferred to C:/tabe/emp.html file, Open it  
in Browser");  
fos.write(b);  
fos.close();  
st.close();  
con.close();  
}
```

23/09/2015

(38)

Ex:- 15

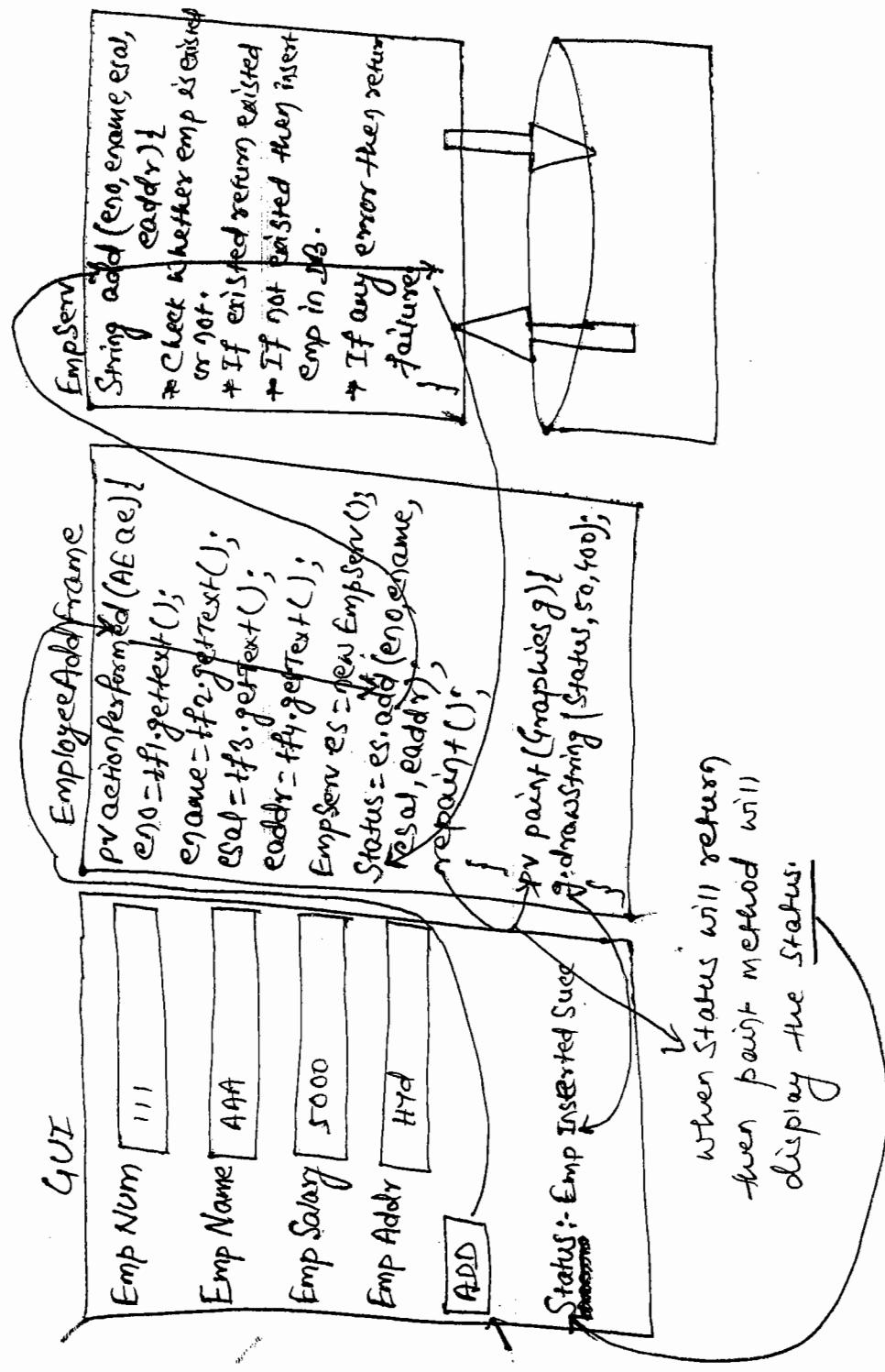


SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

24-09-2015 (40)

Ex:- (16)



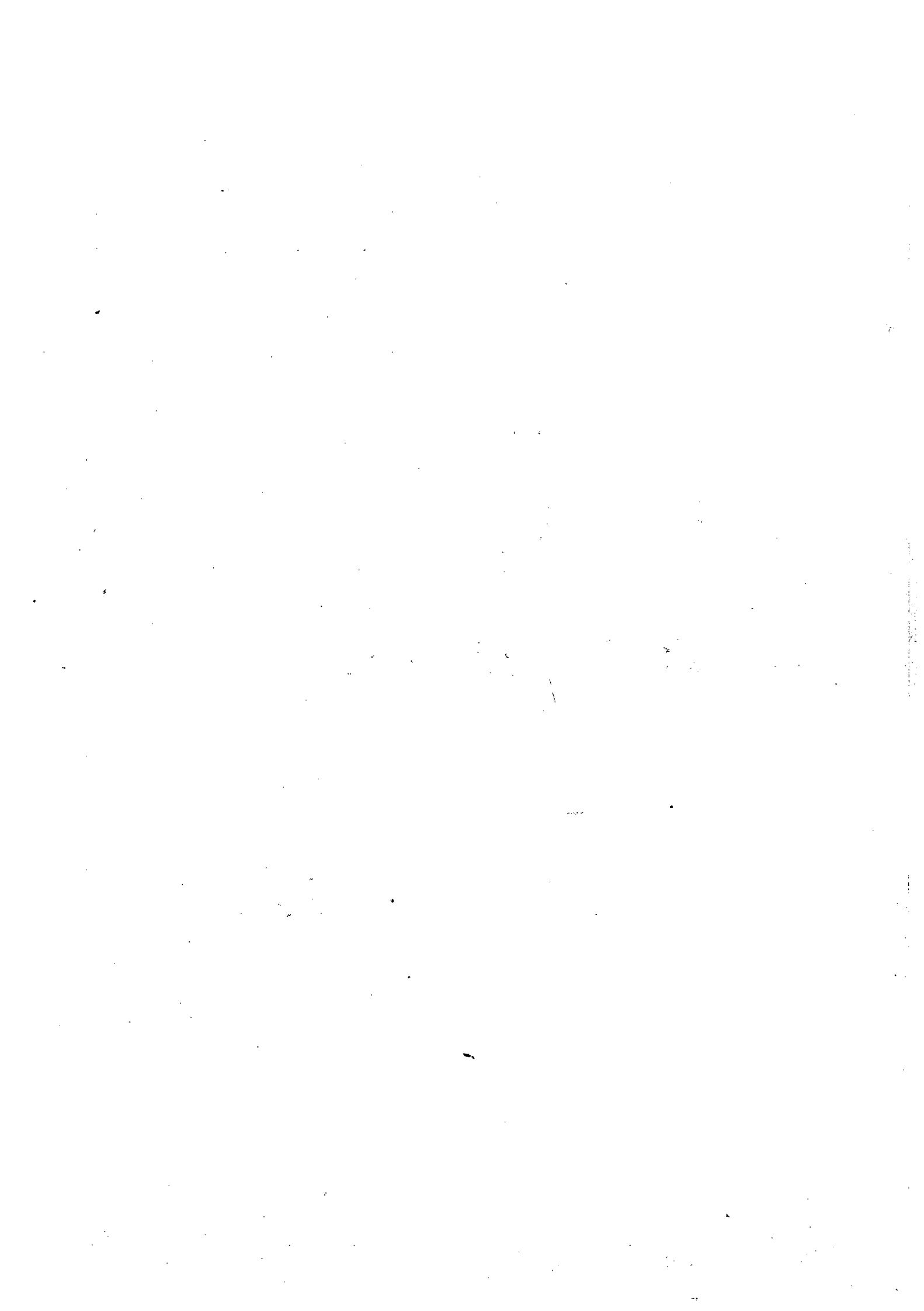


Fig: 17

Fig:

### Student Search frame

### GUI

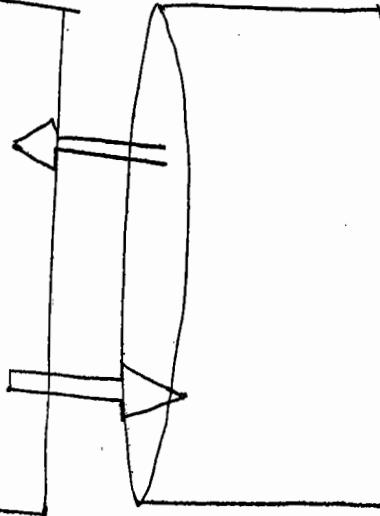
Student ID [S-111]

Search

Std Id = 111  
Std Name = PAA  
Std Addr = H-1d  
or  
Std not existed

```
pv actionPerformed(ActionEvent e) {  
    StdId = tf1.getText();  
    StdServ ss = new StdServ();  
    Std = ss.search(StdId);  
    bp.paint();  
}  
  
pv paint(Graphics g) {  
    if (Std != null) {  
        g.drawString("Std not existed");  
    } else {  
        g.drawString("Std Details");  
    }  
}
```

```
StartToSearch(String std){  
    Check whether std is existed  
    Or not in DB.  
    If std is existed, return std  
    details in form of std  
    If std is not existed then  
    return null value.  
}
```



(42)

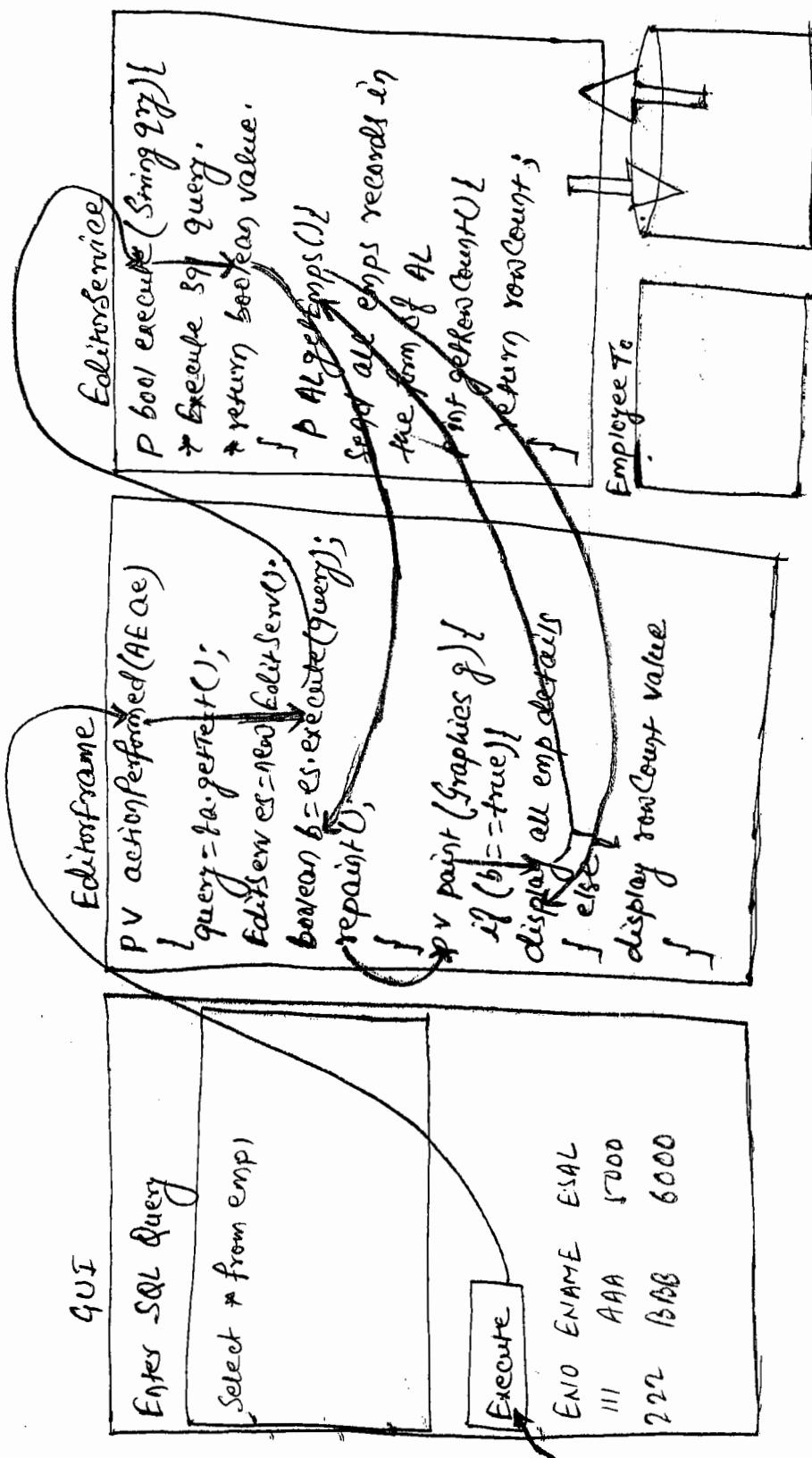
25-09-2015

Program: (17)

25-09-2015

(48)

Ex: (18)



**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

**JdbcApp15:**

```
package com.durgasoft;
import java.awt.Button;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.GraphicsConfiguration;
import java.awt.HeadlessException;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
public class LoginFrame extends Frame implements ActionListener {
    Label l1,l2;
    TextField tf1,tf2;
    Button b1;
    String status="";
    public LoginFrame(){
        this.setVisible(true);
        this.setSize(500,500);
        this.setTitle("Login Frame");
        this.setBackground(Color.green);
        this.setLayout(new FlowLayout());
        this.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
        l1=new Label("User Name");
        l2=new Label("Password");
        tf1=new TextField(20);
        tf2=new TextField(20);
        tf2.setEchoChar('*');
```

```
b1=new Button("Login");
b1.addActionListener(this);

Font f=new Font("arial",Font.BOLD,20);
l1.setFont(f);
l2.setFont(f);
tf1.setFont(f);
tf2.setFont(f);
b1.setFont(f);

this.add(l1);
this.add(tf1);
this.add(l2);
this.add(tf2);
this.add(b1);

}

public void actionPerformed(ActionEvent ae) {
    String uname=tf1.getText();
    String upwd=tf2.getText();
    UserService us=new UserService();
    status=us.checkLogin(uname,upwd);
    repaint();
}

public void paint(Graphics g){
    Font f=new Font("arial",Font.BOLD,30);
    g.setFont(f);
    g.drawString("Status :" +status,50,250);
}

}

package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class UserService {
    Connection con;
    Statement st;
    ResultSet rs;
    String status="";
    public UserService() {
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            con=DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system", "durga");
            st=con.createStatement();
        } catch (Exception e) {
```

```
        e.printStackTrace();
    }
}

public String checkLogin(String uname, String upwd){
    try {
        rs=st.executeQuery("select * from registered_Users where
            uname='"+uname+"' and upwd='"+upwd+"'");
        boolean b=rs.next();
        if(b==true){
            status="Login Success";
        }else{
            status="Login Failure";
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return status;
}
}
```

```
package com.durgasoft;
public class JdbcApp15 {

    public static void main(String[] args) {
        LoginFrame lf=new LoginFrame();
    }
}
```

### JdbcApp16:

```
package com.durgasoft;
import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.GraphicsConfiguration;
import java.awt.HeadlessException;
```

```
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class EmployeeAddFrame extends Frame implements ActionListener {
    Label l1,l2,l3,l4;
    TextField tf1,tf2,tf3,tf4;
    Button b1;
    String status="";
    public EmployeeAddFrame(){
        this.setVisible(true);
        this.setSize(500, 500);
        this.setTitle("Employee Registration Frame");
        this.setBackground(Color.cyan);
        this.setLayout(null);
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
        l1=new Label("Employee Number");
        l2=new Label("Employee Name");
        l3=new Label("Employee Salary");
        l4=new Label("Employee Address");

        tf1=new TextField(20);
        tf2=new TextField(20);
        tf3=new TextField(20);
        tf4=new TextField(20);

        b1=new Button("ADD");
        b1.addActionListener(this);

        Font f=new Font("arial",Font.BOLD,20);
        l1.setFont(f);
        l2.setFont(f);
        l3.setFont(f);
        l4.setFont(f);
        tf1.setFont(f);
        tf2.setFont(f);
        tf3.setFont(f);
        tf4.setFont(f);
        b1.setFont(f);
    }

    public void actionPerformed(ActionEvent ae) {
        if(ae.getSource()==b1) {
            String eno=tf1.getText();
            String ename=tf2.getText();
            String esalary=tf3.getText();
            String eaddress=tf4.getText();

            if(enumber.length() > 0 & ename.length() > 0 & esalary.length() > 0 & eaddress.length() > 0) {
                status="Data Inserted";
                JOptionPane.showMessageDialog(null,"Data Inserted");
            } else {
                status="Data Not Inserted";
                JOptionPane.showMessageDialog(null,"Data Not Inserted");
            }
        }
    }
}
```

```
l1.setBounds(50, 100, 200, 25);
tf1.setBounds(250, 100, 200, 30);
l2.setBounds(50, 150, 200, 25);
tf2.setBounds(250, 150, 200, 30);
l3.setBounds(50, 200, 200, 25);
tf3.setBounds(250, 200, 200, 30);
l4.setBounds(50, 250, 200, 25);
tf4.setBounds(250, 250, 200, 30);
b1.setBounds(50, 300, 100, 30);
this.add(l1);
this.add(tf1);
this.add(l2);
this.add(tf2);
this.add(l3);
this.add(tf3);
this.add(l4);
this.add(tf4);
this.add(b1);
}
public void actionPerformed(ActionEvent ae)
try {
    int eno=Integer.parseInt(tf1.getText());
    String ename=tf2.getText();
    float esal=Float.parseFloat(tf3.getText());
    String eaddr=tf4.getText();

    EmployeeService es=new EmployeeService();
    status=es.add(eno,ename,esal,eaddr);
    repaint();
}
catch (Exception e) {
    e.printStackTrace();
}
}
public void paint(Graphics g){
    Font f=new Font("arial",Font.BOLD,30);
    g.setFont(f);
    g.drawString("Status:"+status, 50, 400);
}

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

```
public class EmployeeService {  
    Connection con;  
    Statement st;  
    ResultSet rs;  
    String status="";  
  
    public EmployeeService() {  
        try {  
            Class.forName("oracle.jdbc.OracleDriver");  
            con=DriverManager.getConnection  
                ("jdbc:oracle:thin:@localhost:1521:xe", "system", "durga");  
            st=con.createStatement();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
    public String add(int eno, String ename, float esal, String eaddr){  
        try {  
            rs=st.executeQuery("select * from emp1 where enno="+eno);  
            boolean b=rs.next();  
            if(b==true){  
                status="Employee Existed Already";  
            }else{  
                st.executeUpdate("insert into emp1  
                    values("+eno+","+ename+","+esal+","+eaddr+")");  
                status="Employee Registration Success";  
            }  
        } catch (Exception e) {  
            status="Employee Registration Failure";  
            e.printStackTrace();  
        }  
        return status;  
    }  
  
    package com.durgasoft;  
  
    public class JdbcApp16 {  
  
        public static void main(String[] args) {  
            EmployeeAddFrame f=new EmployeeAddFrame();  
        }  
    }
```

```
}
```

### JdbcApp17:

```
package com.durgasoft;
import java.awt.Button;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.GraphicsConfiguration;
import java.awt.HeadlessException;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class StudentSearchFrame extends Frame implements ActionListener {
    Label l1;
    TextField tf1;
    Button b1;
    Studentsto;
    public StudentSearchFrame() {
        this.setVisible(true);
        this.setSize(500, 500);
        this.setTitle("Student Search Frame");
        this.setLayout(new FlowLayout());
        this.setBackground(Color.green);
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
        l1=new Label("Student Id");
        tf1=new TextField(20);
        b1=new Button("Search");
        b1.addActionListener(this);
    }
}
```

```
Font f=new Font("arial",Font.BOLD,20);
l1.setFont(f);
tf1.setFont(f);
b1.setFont(f);

this.add(l1);
this.add(tf1);
this.add(b1);
}

public void actionPerformed(ActionEvent arg0) {
    String sid=tf1.getText();
    StudentService ss=new StudentService();
    sto=ss.search(sid);
    repaint();
}

public void paint(Graphics g){
    Font f=new Font("arial",Font.BOLD,30);
    g.setFont(f);
    if(sto==null){
        g.drawString("Student Not Existed", 50, 300);
    }else{
        g.drawString("Student Id : "+sto.getId(), 50, 250);
        g.drawString("Student Name : "+sto.getName(), 50, 300);
        g.drawString("Student Address : "+sto.getAddress(), 50, 350);
    }
}

package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class StudentService {
    Connection con;
    Statement st;
    ResultSet rs;
    StudentTo sto;
    public StudentService() {
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:oci8:@xe", "system",
"durga");
            st=con.createStatement();
        } catch (Exception e) {
```

```
        }
    }

public StudentTo search(String sid){
    try {
        rs=st.executeQuery("select * from student where sid='"+sid+"'");
        boolean b=rs.next();
        if(b==true){
            sto=new StudentTo();
            sto.setSid(rs.getString(1));
            sto.setSname(rs.getString(2));
            sto.setSaddr(rs.getString(3));
        }else{
            sto=null;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return sto;
}

}
```

```
package com.durgasoft;
public class StudentTo {
    private String sid;
    private String sname;
    private String saddr;
    public String getSid() {
        return sid;
    }
    public void setSid(String sid) {
        this.sid = sid;
    }
    public String getSname() {
        return sname;
    }
    public void setSname(String sname) {
        this.sname = sname;
    }
    public String getSaddr() {
        return saddr;
    }
    public void setSaddr(String saddr) {
        this.saddr = saddr;
    }
}
```

```
}

}

package com.durgasoft;

public class JdbcApp17 {

    public static void main(String[] args) {
        StudentSearchFrame sf=new StudentSearchFrame();

    }

}
```

## JdbcApp18

```
package com.durgasoft;
import java.awt.Button;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Label;
import java.awt.TextArea;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.ArrayList;

public class EditorFrame extends Frame implements ActionListener {
    Label l;
    TextArea ta;
    Button b;
    EditorService es;
    boolean bol;
    public EditorFrame(){
        this.setVisible(true);
        this.setSize(700, 800);
        this.setTitle("SQL Editor Frame");
        this.setBackground(Color.green);
        this.setLayout(new FlowLayout());
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e){
                System.exit(0);
            }
        });
    }
}
```

```
        }

    });

l=new Label("Provide SQLQuery");
ta=new TextArea(3,30);
b=new Button("Execute");
b.addActionListener(this);

Font f=new Font("arial",Font.BOLD,20);
l.setFont(f);
ta.setFont(f);
b.setFont(f);

this.add(l);
this.add(ta);
this.add(b);

}

public void actionPerformed(ActionEvent ae){
    String query=ta.getText();
    es=new EditorService();
    bol=es.execute(query);
    repaint();
}

public void paint(Graphics g){
    Font f=new Font("arial",Font.BOLD,30);
    g.setFont(f);
    if(bol==true){
        ArrayList al=es.getEmps();
        g.drawString("ENO ENAME ESAL EADDR",50,200);
        g.drawString("-----", 50, 240);
        int y=300;
        for(int i=0;i<al.size();i++){
            EmployeeTo eto=(EmployeeTo)al.get(i);
            g.drawString(eto.getEno()+" "+eto.getEname()+" "+eto.getEsal()+" "+eto.getEaddr(), 50, y);
            y=y+50;
        }
    }else{
        int rowCount=es.getRowCount();
        g.drawString("Row Count :"+rowCount, 50, 300);
    }
}
```

```
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

public class EditorService {
    Connection con;
    Statement st;
    ResultSet rs;
    ArrayList al;
    boolean bol;
    int rowCount;
    public EditorService() {
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:oci8:@xe", "system",
                "durga");
            st=con.createStatement();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public boolean execute(String sql_Query){
        try {
            bol=st.execute(sql_Query);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bol;
    }
    public ArrayList getEmps(){
        al=new ArrayList();
        try {
            rs=st.getResultSet();
            while(rs.next()){
                EmployeeTo eto=new EmployeeTo();
                eto.setEno(rs.getInt(1));
                eto.setEname(rs.getString(2));
                eto.setEsal(rs.getFloat(3));
                eto.setEaddr(rs.getString(4));
                al.add(eto);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return al;
    }
}
```

```
        }
    }

    catch (Exception e) {
        e.printStackTrace();
    }
    return al;
}
public int getRowCount(){
    try {
        rowCount=st.executeUpdate();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return rowCount;
}

}

package com.durgasoft;

public class EmployeeTo {
    private int eno;
    private String ename;
    private float esal;
    private String eaddr;
    public int getEno() {
        return eno;
    }
    public void setEno(int eno) {
        this.eno = eno;
    }
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public float getEsal() {
        return esal;
    }
    public void setEsal(float esal) {
        this.esal = esal;
    }
    public String getEaddr() {
        return eaddr;
    }
}
```

```
        }
        public void setEaddr(String eaddr) {
            this.eaddr = eaddr;
        }
    }

package com.durgasoft;

public class JdbcApp18 {

    public static void main(String[] args) {
        EditorFrame f=new EditorFrame();
    }
}
```

## ResultSet Types:

In jdbc applications ResultSets can be divided into two types:

-->As per the ResultSet concurrency there are two types of ResultSets.

### 1) Read only ResultSet

It is a ResultSet object, it will allow the users to read the data only.

To represent this ResultSet object ResultSet interface has provided the following constant

```
public static final int CONCUR_READ_ONLY
```

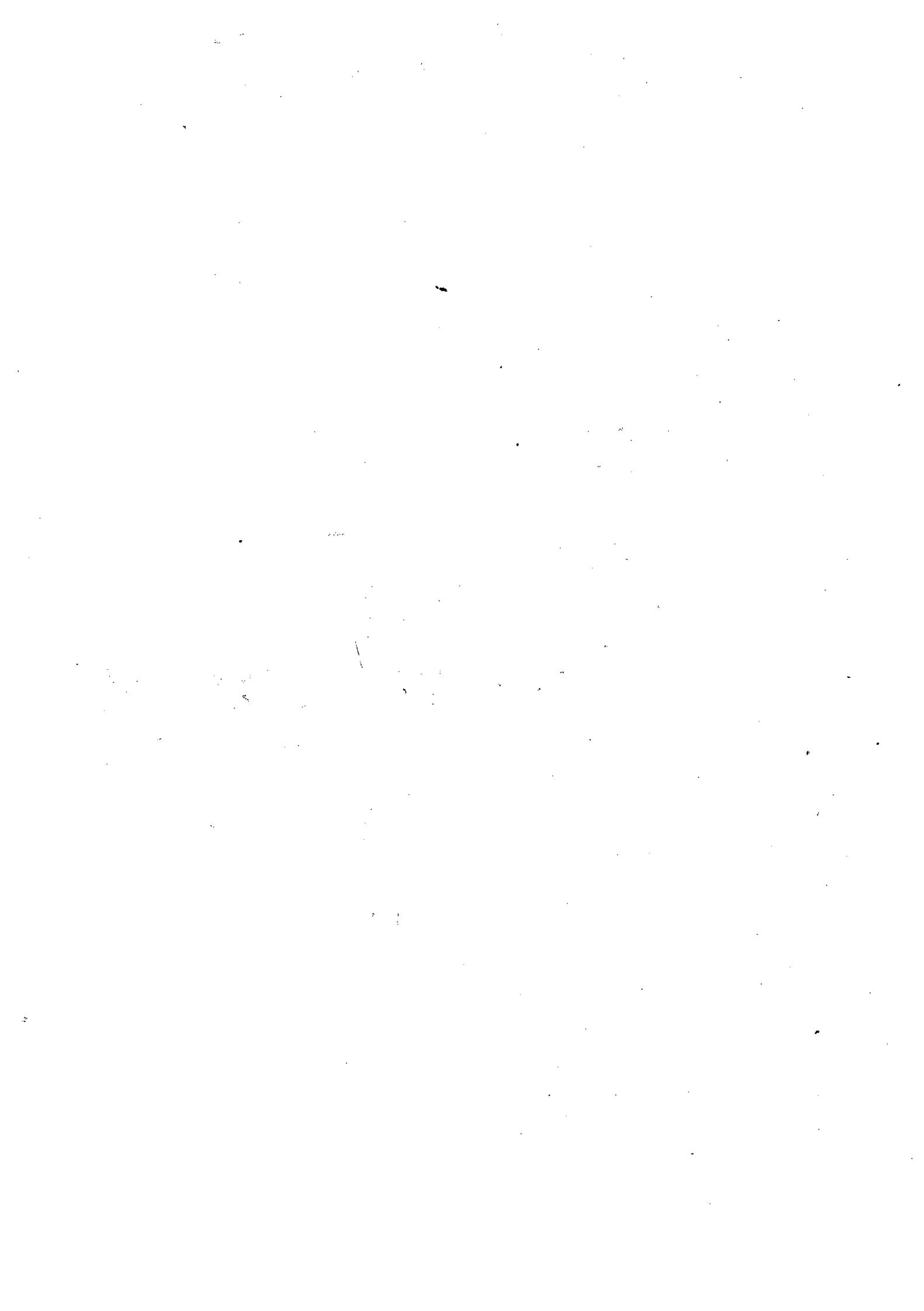
### 2) Updatable ResultSet:

It is a ResultSet object, it will allow the users to perform updations on its content.

To represent this resultset object ResultSet interface has provided the following constant.

```
public static final int CONCUR_UPDATABLE
```

-->As per the ResultSet cursor movement there are two types of ResultSets



03/10/2015

## Metadata:-

Data about the data is called as Metadata.

There are two types of metadata in JDBC.

1) Database Metadata

2) Resultset Metadata

### 1) Database Metadata:-

Data about database is called as Database Metadata.

Database Metadata is able to include the details of the database like database name, database version name, supported SQL keywords, supported String functions, supported numeric functions, ...

To represent Database Metadata Object JDBC API has provided an interface in the form of

`java.sql.DatabaseMetadata`

To get DatabaseMetadata Object we have to use the following method from java.sql.Connection.

`public DatabaseMetadata getMetaData() throws SQLException`

To get database metadata like database name, database product version, driver major version and minor version, supported SQL keywords, String functions... We have to use the following methods.

`public String getDatabaseProductName() throws SQLException`  
`public String getDatabaseProductVersion() throws SQLException`  
`public int getDriverMajorVersion() throws SQLException`  
`public int getDriverMinorVersion() throws SQLException`  
`public String getStringFunctions() throws SQLException`  
`public String getNumericFunctions() throws SQLException`

(46)

```

public boolean supportBatchUpdates() throws SQLException;
public boolean supportStoredProcedures() throws SQLException;
public String getURL() throws SQLException;
public String getUsername() throws SQLException;
public String getSQLKeywords() throws SQLException.

```

Ex:- (44)

```

import java.sql.*;
public class Job6cApp44 {
    public void main(String[] args) throws Exception {
        Class.forName("Oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "System", "durga");
        DatabaseMetaData md = con.getMetaData();
        System.out.println(md.getDatabaseProductName());
        System.out.println(md.getDatabaseProductVersion());
        System.out.println(md.getDriverMajorVersion());
        System.out.println(md.getDriverMinorVersion());
        System.out.println(md.getSQLKeywords());
        System.out.println(md.getStringFunctions());
        System.out.println(md.getNumericFunctions());
        System.out.println(md.supportBatchUpdates());
        System.out.println(md.supportStoredProcedures());
        System.out.println(md.getURL());
        System.out.println(md.getUserName());
    }
}

```

## ② ResultSetMetadata:-

- Data about ResultSet object is called as ResultSetMetadata.
- In JDBC, ResultSetMetadata is able to include the details like no. of columns, Column names, Column data types, Column sizes, ...
- To represent ResultSetMetadata Object in JDBC application, JDBC API has provided a predefined interface in the form of java.sql.ResultSetMetadata.

To get ResultSetMetadata Object in JDBC applications, we have to use the following method from java.sql.ResultSet interface.

```
public ResultSetMetadata getMetaData() throws SQLException.
```

- To get no. of columns which are available in ResultSet object we have to use the following method

```
public int getColumnCount() throws SQLException
```

To get name of a particular column we have to use the following method.

```
public String getColumnName(int column_index) throws SQLException
```

- To get datatype of a particular column we have to use the following method.

```
public String getColumnTypeName(int column_index) throws SQLException
```

- To get size of a particular column we have to use the following method.

```
public int getColumnDisplaySize(int column_index) throws SQLException.
```

Ex:- (45)

```

import java.sql.*;
public class jdbcApp45{
    public static void main(String[] args) throws Exception {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "durga");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from emp");
        ResultSetMetaData md = rs.getMetaData();
        int count = md.getColumnCount();
        System.out.println("No. of Columns :" + count);
        System.out.println();
        for (int i = 1; i <= count; i++) {
            System.out.println("Column Name :" + md.getColumnName(i));
            System.out.println("Column Size :" + md.getColumnDisplaySize(i));
            System.out.println("-----");
        }
        st.close();
        con.close();
    }
    System.out.println("Column Datatype :" + md.getColumnTypeName(i));
}

```

Ex:- (26)

```
import java.sql.*;
public class JDBCAPP26{
    public void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:
thin:@localhost:1521:xe","system","deurga");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("Select * from emp");
        ResultSetMetaData md = rs.getMetaData();
        int count = md.getColumnCount();
        for (int i=1; i<=count; i++)
        {
            System.out.println(md.getColumnName(i)+"|");
        }
        System.out.println();
        System.out.println("-----");
        while(rs.next())
        {
            for (int j=1; j<=count; j++)
            {
                System.out.print(rs.getString(j)+"|");
            }
            System.out.println();
        }
        st.close();
        con.close();
    }
}
```

## MySQL:- (Command prompt):-

- > connect;
- > Show databases; → To see databases which are already created in database.
- > drop database durgadb; → durga is removed from the particular database.
- > Create database durgadb; The drop db will be created.
- > Show databases
- > Use durgadb; → Database changed.
- > Show tables; → As no table is created till now, it is showing empty.
- > Create table emp1 (eno int(5), Primary key, ename char(10), esal float, eaddr Char(10));
- > desc emp1;
- > insert into emp1 values (11, 'AAA', 5000, "Hyd");  
 { 222, 'BBB', 6000, "Hyd"; }  
 { 333, 'CCC', 7000, "Hyd"; }  
 { 444, 'DDD', 8000, "Hyd"; } } → To insert values into the table.
- > select \* from emp1;
- > update emp1 set esal = esal + 500 where esal < 10000;
- > Select \* from emp1;
- > delete from emp1 where esal < 10000;
- > select \* from emp1;
- > drop table emp1;
- > show tables;
- > drop database durgadb;
- > Show databases;
  


---

- >> drop table emp1;
- > Create table emp1 (eno int(5), Primary key, ename char(10), esal float, eaddr char(10));
- > Commit;

### Type-4 driver provided by MySQL:-

- Driver Class: com.mysql.jdbc.Driver
- Driver URL: jdbc:mysql://localhost:3306/db-name.
- mysql-connector-java-5.1.6.jar.

In JDBC applications, if we want to use MySQL database then we have to use the above Driver class name and driver URL.

MySQL database has provided "com.mysql.jdbc.Driver" class in the form of mysql-connector-java-5.1.6.jar, so that, we have to set "classpath" environment variable to mysql-connector-java-5.1.6.jar file location.

Set classpath = D:\Software\Datasets\MySQL\MySQL 5.1.6\mysql-connector-java-5.1.6.jar;

Ex:- ④7

```
import java.sql.*;
public class JDBCApp47
{
    public void main(String[] args) throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://
            localhost:3306/deergads", "root", "root");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("Select * from emp");
        ResultSetMetaData md = rs.getMetaData();
        int count = md.getColumnCount();
        for(int i=1; i<=count; i++)
        {
            System.out.println(md.getColumnName(i) + " | ");
        }
    }
}
```

```

S.o.PIN();
S.o.PIN("-----");
while(rs.next()){
    for(i= i=1; i<=count; i++){
        S.o.P(rs.getString(i)+ "12");
    }
    S.o.PIN();
}
S.o.Close();
con.Close();
}

```

≡

### ResultSet:-

In JDBC applications, ResultSet objects are classified into the following two ways.

→ On the basis of the ResultSet Concurrency there are two types of ResultSets.

#### (a) ReadOnly ResultSet :-

It able to allow only to read data, not to allow modifications on its content.

To represent this ResultSet Object, ResultSet interface has provided the following constant:

→ public static final int CONCUR\_READ\_ONLY;

#### (b) Updatable ResultSet :-

It able to allow to perform modifications on its content.

To represent this ResultSet Object, ResultSet interface has provided the following Constant:

→ public static final int CONCUR\_UPDATABLE;

② On the basis of the Resultset Cursor moment, there are two types of Resultsets.

① Forward Only Resultset:-

- It able to allow to retrieve data in only forward direction.
- To represent this ResultSet Object, Resultset interface has provided the following Constant.

public static final int TYPE\_FORWARD\_ONLY;

② Scorable Resultset:-

These Resultset Objects are able to allow to read data in both forward direction and backward direction.

There are two types of Scorable Resultset Objects.

① Scroll Sensitive Resultset:-

These Resultset Objects are allowing later database modifications directly.

To represent this Resultset Object, Resultset interface has provided the following Constant.

public static final int TYPE\_SCROLL\_SENSITIVE;

② Scroll Inensitive Resultset:-

These Resultset Objects are not allowing later database modifications.

- To represent this Resultset Object, Resultset interface has provided the following Constant.

public static final int TYPE\_SCROLL\_INSENSITIVE;

- The default Resultset type in JDBC Applications is "Read Only and forward Only" Resultset.

→ If we want to create ResultSet Object with a particular type then we have to specify the ResultSet types [Constants] at the time of creating Statement Object, for this we have to use the following method.

`public Statement createStatement (int forward_only/  
scrollSensitive/scroll_inensitive, int Read_only/updatable)  
throws SQLException;`

Ex:- `Statement st = con.createStatement (ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);`  
`ResultSet rs = st.executeQuery ("Select * from emp1");`

- In JDBC applications, by using Scrollable ResultSet Objects we are able to retrieve data in both forward direction and backward direction.
- To retrieve data in forward direction we are able to use the following methods.

`public boolean next()`  
`public xxx getxxx (int Column_index)`  
or  
`public xxx getxxx (String Column_Name)`  
where xxx may be byte, short, int, ...

- If we want to retrieve data in backward direction then we have to use the following steps for each and every record.

① Check whether previous record is available or not from the current cursor position. If it is available then move cursor to the previous record position.

To perform the above action, we have to use the following method.

`public boolean previous () throws SQLException`

- It will check whether previous record is available or not if it is available then it will return "true" value, if it is not available, then it will return "false" value.

② After getting Resultset cursor to the record position then retrieve data from the columns individually by using the following methods.

public xxx getxxx(int column-index) throws SQLException

or

public xxx getxxx(String column-name) throws SQLException

where xxx may be byte, short, int, ...

→ To move cursor from one record to another record Resultset interface has provided the following methods.

① public void beforefirst() throws SQLException

→ It will move cursor to before first record position.

② public void afterlast () throws SQLException

→ It will move cursor to after last record position.

③ public boolean first() throws SQLException

→ It will move cursor to first record position.

④ public boolean last() throws SQLException.

→ It will move cursor to last record position.

⑤ public boolean absolute(int rec-position) throws SQLException

→ It will move cursor to the specified record position, where if we provide +ve value of rec-position then records are calculated from first record and cursor moves in forward direction. If rec-position value is -ve value then records position is calculated from last record in backward direction.

⑥ public boolean relative(int no-of-records)

→ It will skip the cursor over the specified no. of records. If the no. of records value is +ve then cursor will be skipped

(51)

Out in forward direction, from current Cursor position. If the no. of records value is -ve then cursor will be skipped  
Out in backward direction (from current Cursor position)

Ex:- JdbcApp19 :-

Statement St =

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

program :- JdbcApp19 :-

```
import java.sql.*;
public class JdbcApp19 {
    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/olurgaab", "root", "root");
        Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                         ResultSet.CONCUR_UPDATABLE);
        ResultSet rs = st.executeQuery("select * from emp1");
        System.out.println("Data Forward Direction");
        System.out.println("ENO|ENAME|ESAL|EADDR|");
        System.out.println("-----");
        while(rs.next()){
            System.out.println(rs.getInt(1)+"|"+rs.getString(2)+"|"+rs.getFloat(3)+"
                           "+rs.getString(4));
        }
        System.out.println("Data Backward Direction");
        System.out.println("ENO|ENAME|ESAL|EADDR|");
        System.out.println("-----");
        while(rs.previous()){
            System.out.println(rs.getInt(1)+"|"+rs.getString(2)+"|"+rs.getFloat(3)+"
                           "+rs.getString(4));
        }
        con.close();
    }
}
```

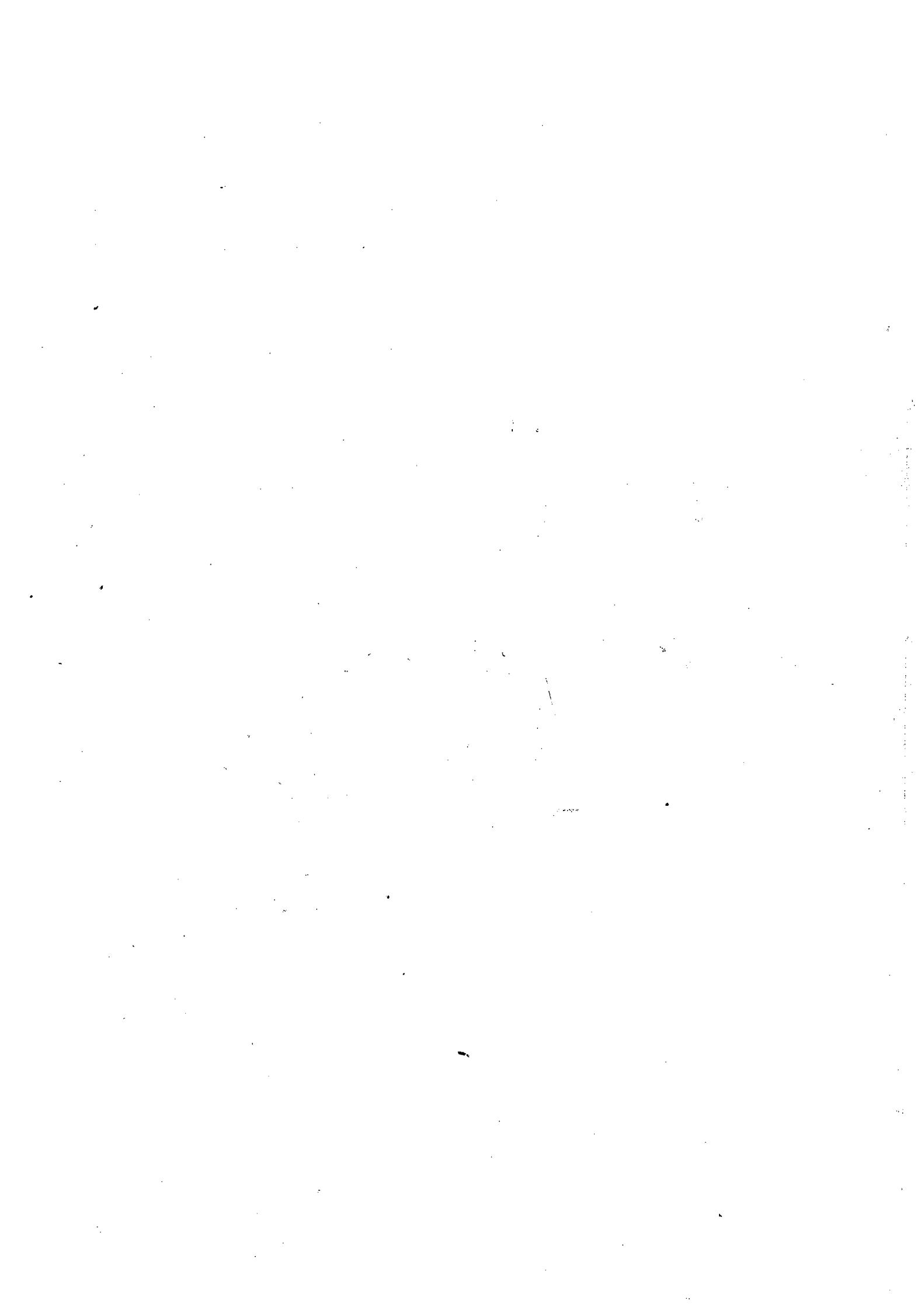
====

Ex:- JdbcApp20 :-

```

import java.sql.*;
public class JdbcApp20{
    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/algudbs", "root", "root");
        Statement st = con.createStatement("ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE");
        ResultSet rs = st.executeQuery("Select * from emp");
        rs.last();
        System.out.println(rs.getInt(1));
        rs.previous();
        System.out.println(rs.getInt(1));
        rs.last();
        System.out.println(rs.getInt(1));
        rs.first();
        System.out.println(rs.getInt(1));
        rs.absolute(3);
        System.out.println(rs.getInt(1));
        rs.absolute(-3);
        System.out.println(rs.getInt(1));
        rs.first();
        System.out.println(rs.getInt(1));
        rs.relative(2);
        System.out.println(rs.getInt(1));
        rs.last();
        System.out.println(rs.getInt(1));
        rs.relative(-2);
        System.out.println(rs.getInt(1));
    }
}

```



## JdbcApp21:

```
package com.durgasoft;

import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class PlayerFrame extends Frame implements ActionListener {
    Button b1,b2,b3,b4;
    String label;
    EmployeeTo eto;
    EmployeeService es;
    public PlayerFrame() {
        this.setVisible(true);
        this.setSize(500, 500);
        this.setTitle("Player Frame");
        this.setBackground(Color.green);
        this.setLayout(null);
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
        b1=new Button("First");
        b2=new Button("Next");
        b3=new Button("Previous");
        b4=new Button("Last");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        b4.addActionListener(this);
        Font f=new Font("arial",Font.BOLD,20);
        b1.setFont(f);
        b2.setFont(f);
        b3.setFont(f);
        b4.setFont(f);
```



```
package com.durgasoft;

public class JdbcApp21 {

    public static void main(String[] args) {
        PlayerFrame pf=new PlayerFrame();
    }

}
```

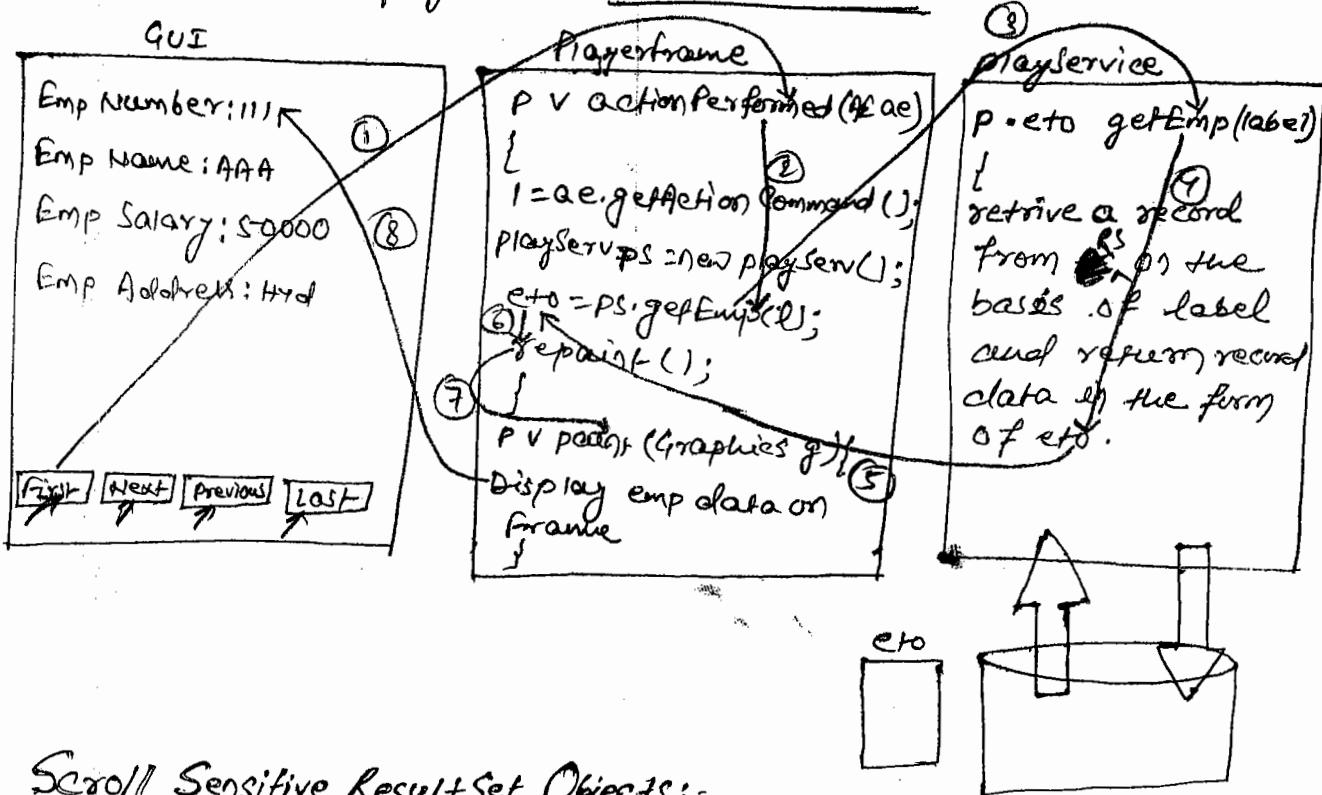
#### Scroll Sensitive ResultSet:

```
JdbcApp22
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class JdbcApp22 {
    public static void main(String[] args) throws Exception {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con=DriverManager.getConnection("jdbc:odbc:nag","system","durga");
        Statement st=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
        ResultSet rs=st.executeQuery("select * from emp1");
        System.out.println("Data Before Updations");
        System.out.println("ENO    ENAME    ESAL    EADDR");
        System.out.println("-----");
        while(rs.next()){
            System.out.println(rs.getInt(1)+"    "+rs.getString(2)+"    "+rs.getFloat(3)+""
            "+rs.getString(4));
        }
        System.out.println("Application is in Pausing state, please update database");
        System.in.read();
        rs.beforeFirst();
        System.out.println("Data After Updations");
        System.out.println("ENO    ENAME    ESAL    EADDR");
```



JdbcApp21 → for program → Refer Text book



### Scroll Sensitive ResultSet Objects:-

In JDBC Applications, Scroll sensitive ResultSet Objects are able to allow the later database modifications automatically.

→ To prove Scroll sensitive ResultSet Objects in JDBC Applications then we have to use the following steps.

- ① Get Scroll Sensitive ResultSet Object.
- ② Display data from scroll Sensitive ResultSet Object.
- ③ Paste JDBC Application execution by using System.in.read() method.
- ④ Go to database table and perform modifications on the respective table and perform Commit Operation.
- ⑤ Come back to the JDBC Application and refresh the data from the same ResultSet Object by refreshing each and every record.

To Refresh Current Row we have to use the following method.

public void refreshRow()

Check whether the data before modifications and after modifications is same then the Resultset Object is not Scroll Sensitive Resultset Object. If the data before modifications and after modifications are different then the Resultset Object is Scroll sensitive Resultset.

Ex:- (22)

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

In jdb applications, Scroll-Sensitive Resultset Object are supported by Type-1 driver provided by SUN MICRO SYSTEMS and Type-4 driver provided by MySQL. Scroll-Sensitive Resultset Object is not supported by Type-4 driver provided by Oracle. If we execute the above application with Type-4 driver provided by Oracle then JVM will raise an exception like "java.sql.SQLException: Unsupported feature: refreshRow;"

- In jdb applications, Scroll-Insensitive Resultset Object is not supported by Type-1 driver provided by SUN MICRO SYSTEMS, Type-4 driver provided by Oracle and MySQL Databases.
- In jdb, by using Updatable Resultset Objects, we are able to perform the database operations like insert, update and delete without using the respective SQL queries like, insert, update and delete.
- If we want to insert a record into database table by using Updatable Resultset Object then we have to use the following steps:-

### ① Create Updatable Resultset Object:-

```
Statement st = con.createStatement (Resultset.TYPE_SCROLL_SENSITIVE,
                                Resultset.CONCUR_UPDATABLE);
Resultset rs = st.executeQuery ("Select * from emp1");
```

### ② Move Cursor to end of the Resultset Object and take a buffer to store new record data:

- To perform this action, we have to use the following methods.
  - public void moveToInsertRow() throws SQLException.

[Or]

public void afterLast() throws SQLException.

③ Insert new Record data in Updatable ResultSet Object:-

→ To insert a particular column value in new record we have to use the following method.

public void updateXXX(~~int column index, xxx value~~) throws SQLException.

Ex:- rs.updateInt(1, 111);

rs.updateString(2, "AAA");

rs.updateFloat(3, 50000.0f);

④ Move new Record data from Updatable ResultSet Object to database table:-

→ To perform this action we have to use the following method.

public void insertRow() throws SQLException.

Ex:- rs.insertRow();

Statement st = con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE, ResultSet.CONCUR\_UPDATABLE);  
ResultSet rs = st.executeQuery("Select \* from emp1");

rs.moveToInsertRow();

rs.updateInt(1, 444);

rs.updateString(2, "DDD");

rs.updateFloat(3, 8000);

rs.insertRow();

ENO	ENAME	ESAL
111	AAA	5000
222	BBB	6000
333	CCC	7000
444	DDD	8000

ENO	ENAME	ESAL
111		
222		
333		
444	DDD	8000

Ex:- (23) → refer page NO. 53, 2) Text Book.

→ Update DBTable by Using Updatable Resultset Object :-

→ If we want to perform operations on database table by using Updatable Resultset Objects then we have to use the following steps:-

(1) Create Updatable Resultset Object :-

```
Statement st = con.createStatement (ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
```

```
ResultSet rs = st.executeQuery ("Select * from emp");
```

(2) Perform Operations on Updatable Resultset Object :-

To update a particular column in the present record then we have to use the following method.

```
public void updateXXX (int Column_Index, XXX Value) throws SQLException;
```

Ex:-  
`rs.next();  
 rs.updateFloat(3, 7000);`

(3) Reflect Updates from Resultset Object to Database table :-

To reflect updates from Updatable Resultset to Database table we have to use the following method.

```
public void updateRow();
```

Ex:- `rs.updateRow();`

```
Statement St = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,  
                                ResultSet.CONCUR_UPDATABLE);
```

```
ResultSet rs = St.executeQuery("Select * from emp");
```

```
while(rs.next())  
{  
    float esal = rs.getFloat(3);  
    if(esal < 10000)  
    {  
        esal = esal + 500;  
        rs.updateFloat(3, esal);  
        rs.updateRow();  
    }  
}
```

ENO	ENAME	ESAL
111	AAA	5500 6000 6400
222	BBB	6000 6500 7100
333	CCC	7000 7500

ENO	ENAME	ESAL
111	AAA	5500 6000 6400
222	BBB	6000 6500 7100
333	CCC	7000 7500

In jdbc applications, if we want to delete records from database table by using updatable Resultset Object then we have to use the following steps.

① Get Updatable Resultset Object:-

```
Statement St = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,  
                                ResultSet.CONCUR_UPDATABLE);
```

```
ResultSet rs = St.executeQuery("Select * from emp");
```

② Move Cursor to the record position which record we want to delete and delete record:-

To delete record from Database table we have to use the following method.

```
public void deleteRow();
```

```
Ex:- rs.last();  
     rs.deleteRow();
```

13/10/2015

(56)

Note:- Updatable Resultset Objects are supported by Type-1 driver provided by SUN Microsystems and Type-4 driver's provided by MySQL database, but Updatable Resultset Objects are not supported by Type-4 driver provided by Oracle-J.

Z

### PreparedStatement:-

Q. What is the diff. between Statement and PreparedStatement?

→ In JDBC applications, when we have a requirement to execute all the SQL queries independently then we have to use "Statement".

In JDBC applications, when we have a requirement to execute the same SQL query in the next sequence, where to improve the application performance we have to use "PreparedStatement".

In the above requirement, if we use Statement then Database Engine will perform Query Tokenization, Query Parsing, Query Optimization, and Query Execution at each and every time of executing the same SQL query without having changes, this approach will increase burden to the Database Engine, it will reduce application performance.

In the above context, to improve application performance, we have to use an alternative that is "PreparedStatement".

→ In case of PreparedStatement, Database Engine will perform Query Tokenization, Query Parsing, Query Optimization and Query Execution Only one time for all the SQL queries execution.

Z

## ⑥ Steps to use PreparedStatement in JDBC application:-

If we want to use PreparedStatement in JDBC apps then we have to use the following steps.

### ① Create prepared Statement Object:-

To Create PreparedStatement Object, we have to use the following method from java.lang.Connection.

```
public PreparedStatement prepareStatement(String query-fmt)  
        throws SQLException;
```

Ex:-

```
PreparedStatement pst = con.prepareStatement("insert  
into emp values(?, ?, ?);");
```

Where '?' are called as place Holders or positional parameters.

### Internal flow:-

When JVM encounters the above instruction, JVM will send the specified SQL query format to database engine, where Database Engine will perform Query Tokenization, Query Parsing, Query Optimization and Query Execution and Create a buffer with the Specified positional parameters called as "Query plan" or "Execution plan"; w.r.t the Query plan, JVM will Create PreparedStatement Object of java Application with the same positional parameters.

## (2) Set values to Positional parameters in PreparedStatement Object:-

Object:-

To set values to positional parameters in PreparedStatement Object we have to use few following methods:-



- public void setxxx (int param\_Index, xxx value)  
throws SQLException

Eg:-  
PST.setInt(1, 111);  
PST.setString(2, "AAA");  
PST.setFloat(3, 5000);

When JMY encounters the above instructions, JMY will set the specified values to the positional parameters in PreparedStatement Object and the same values are transferred to positional parameters in Query plan.

## (3) Perform the Database Operation:-

When values coming to positional parameters in query plan then we have to make database engine to pickup the values from query plan and to perform the required database operations.

 [Cont...]

If the SQL query is select SQL query then we have to use the following method.

public ResultSet executeQuery() throws SQLException.

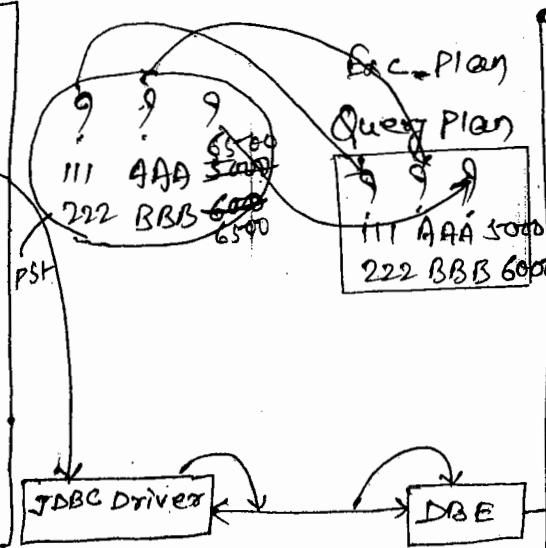
If the SQL query format is Non-Select then we have to use the following method.

public int executeUpdate() throws SQLException.

PreparedStatement pst = con.prepareStatement("insert into emp)  
values(???)");

Java App

```
pst.setInt(1, 111);  
pst.setString(2, "AAA");  
pst.setFloat(3, 5000);  
pst.executeUpdate();
```



EMP		
ENO	ENAME	ESAL
111	AAA	5000
222	BBB	6000

```
pst.setInt(1, 222);  
pst.setString(2, "BBB");  
pst.setFloat(3, 6000);  
pst.executeUpdate();
```

Σ

- Query Tokenization
- Query Parsing
- Query Optimization
- Query Execution.

## JdbcApp23:

The following example demonstrates how to insert no.of records into database table through a Jdbc application

```

package com.durgasoft;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Jdbcapp23 {

    public static void main(String[] args) throws Exception {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con=DriverManager.getConnection("jdbc:odbc:nag","system","durga");
        Statement st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);
        ResultSet rs=st.executeQuery("select * from emp1");
        rs.moveToInsertRow();
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        Scanner s=new Scanner(System.in);

        while(true){
            System.out.print("Employee Number :");
            int eno=Integer.parseInt(br.readLine()); → int eno = s.nextInt();
            System.out.print("Employee Name :");
            String ename=br.readLine(); → String ename = s.next();
            System.out.print("Employee Salary :");
            float esal=Float.parseFloat(br.readLine()); → float esal = s.nextFloat();
            System.out.print("Employee Address :");
            String eaddr=br.readLine(); → String eaddr = s.next();
            rs.updateInt(1, eno);
            rs.updateString(2, ename);
            rs.updateFloat(3, esal);
            rs.updateString(4, eaddr);
            rs.insertRow();

            System.out.println("Employee inserted Successfully");
            System.out.print("Onemore Employee[Yes/no]?");
            String option=br.readLine(); → String option = s.nextLine();
            if(option.equals("no")){
                break;
            }
        }
    }
}

```

*(Handwritten notes and annotations)*

- A large black 'X' is drawn over the entire code block.
- Annotations include:
  - "sun.jdbc.odbc.JdbcOdbcDriver" is crossed out with a large 'X'.
  - "Scanner s=new Scanner(System.in);)" is annotated with "we can use the above Scanner with the place of BufferedReader".
  - "for dynamic input" is written near the "Scanner" annotation.
  - "int eno = s.nextInt();" is written next to the line "int eno=Integer.parseInt(br.readLine());".
  - "String ename = s.next();" is written next to the line "String ename=br.readLine();".
  - "float esal = s.nextFloat();" is written next to the line "float esal=Float.parseFloat(br.readLine());".
  - "String eaddr = s.next();" is written next to the line "String eaddr=br.readLine();".



```

    }st.close();
con.close();
}

}

```

NOTE: In jdbc applications updatable ResultSets could not be supported by Type 4 Driver provided by oracle

--> In jdbc applications by using updatable ResultSet object it is possible to update database

--> To perform this we have to use the following steps

#### Step1: get updatable ResultSet object

Statement

```
st=con.createStatement(resultSet.TYPE_SCROLL_SENSITIVE,resultSet.CONCUR_UPDATABLE);
```

```
ResultSet rs=st.executeQuery("select * from emp");
```

#### Step2: update Resultset Object Temporarily

To achieve this we have to use the following method

```
public void updatexx(int field_Name,xxx value)
```

```
ex: rs.updateFloat(3,7000.0f);
```

#### Step3: Make temporary updation as permanent updation on resultset obejct as well as on database to achieve this we have to use the following method

```
public void updateRow()
```

```
Ex:rs.updateRow();
```

```
JdbcApp24:
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class Jdbcapp24 {
```

(How to update database table  
without using SQL Query?)

```

public static void main(String[] args) throws Exception {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); → MySQL driver
    Connection con=DriverManager.getConnection("jdbc:odbc:nag", "system", "durga");
    Statement st=con.createStatement	ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATABLE);

    ResultSet rs=st.executeQuery("select * from emp1"); → Using sql query
    while(rs.next()){
        float esal=rs.getFloat(3);
        if(esal<10000){
            float new_Sal=esal+500; → esal = esal + 500
            rs.updateFloat(3, new_Sal);
            rs.updateRow();
        }
    }
    st.close();
    con.close();
}
}

```

-->In Jdbc applications by using updatableResultSet object it is possible to delete records on database table to achieve this we have to use the following method

**public void deleteRow()**

Ex:rs.deleteRow();

```

JdbcApp25:
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Jdbcapp25 {
    public static void main(String[] args) throws Exception {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection
        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "durga"
        );
        Statement st=con.createStatement	ResultSet.TYPE_SCROLL_SENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
        ResultSet rs=st.executeQuery("select * from emp1");
    }
}

```

```

rs.last();
rs.deleteRow();
st.close();
con.close();
}

}

while(rs.next()){
int eno=rs.getInt(1);
if(eno==888){
rs.deleteRow();
System.out.println("Employee Deleted successfully");
}
st.close();
con.close();
}

```

-->To move ResultSet cursor to a particular Record position we have to use the following method from resultset

**public void absolute(int position)**

-->To move ResultSet cursor over some no.of records we have to use the following method from Resultset

**public void relative(int no.of.records)**

-->If we have bulk of records in database table,where if we are trying to retrieve all the records at a time into resultset object automatically Jdbc application performance will be reduced.

-->In the above context to improve the performance of Jdbc application we have to fetch the limited no. of records in multiple attempts.

-->To specify the no.of records which we want to fetch at an attempt then we have to use the following method

**public void setFetchSize(int size)**

-->To get the specified fetch size value from resultset we have to use the following method

**public int getFetchSize()**

\* Prepared Statement:-

What is the difference between Statement and PreparedStatement?

Ans:

In Jdbc applications,when we have a requirement to execute the SQL queries independently,we have to use Statement.

In Jdbc applications,when we have a requirement to execute same SQL query in the next sequence where to improve the performance of Jdbc applications,we have to use PreparedStatement.



3. Make Database engine to pickup the values from Query plan and perform the respective operations per the generalised SQL query which we provided

If the generalised SQL query belongs to selection group, then we have to use the following method.

```
public ResultSet executeQuery() throws Exception
```

If the generalised SQL query belongs to updation group, then we have to use the following method

```
public int executeUpdate() throws SQLException
```

```
Eg: int rowCount=pst.executeUpdate();
```

```
JdbcApp26:
package com.durgasoft;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class JdbcApp26 {

    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/durgadb","root","root");
        PreparedStatement pst=con.prepareStatement("insert into emp1 values(?, ?, ?, ?)");
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        while(true){
            System.out.print("Employee Number :");
            int eno=Integer.parseInt(br.readLine());
            System.out.print("Employee Name :");
            String ename=br.readLine();
            System.out.print("Employee Salary :");
            float esal=Float.parseFloat(br.readLine());
            System.out.print("Employee Address :");
            String eaddr=br.readLine();
            pst.setInt(1, eno);
            pst.setString(2, ename);
            pst.setFloat(3, esal);
            pst.setString(4, eaddr);
            pst.executeUpdate();
        }
    }
}
```

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

*Cf with scanner we can use next() method for getting the op/*

*I + will work Replace with get in Java*

*Console c=System.console();*

*int echo=Integer.parseInt(c.readLine("Employee Number :"));*

Note:- `Console c = System.console();`  
↳ It is not working in java 8, it will work in java 6.

3U

DURGASOFT

JDBC

MR.NAGOORBABU

```
System.out.println("Employee Inserted Successfully");
System.out.print("One more Employee(yes/no) :");
String option=br.readLine();
if(option.equals("no")){
    break;
}
con.close();
}}
```

JdbcApp27

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

public class JdbcApp27 {

    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/durgadb","root","root");
        PreparedStatement pst=con.prepareStatement("update emp1 set esal=? where esal=?");
        pst.setInt(1, 500);
        pst.setFloat(2, 10000.0f);
        int rowCount=pst.executeUpdate();
        System.out.println("Records Updated : "+rowCount);
        con.close();
    }
}
```

JdbcApp28:

```
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class JdbcApp28 {
    public static void main(String[] args) throws Exception{
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/durgadb","root","root");
        PreparedStatement pst=con.prepareStatement("select * from emp1 where esal=?");
    }
}
```

```

pst.setFloat(1, 10000.0f);
ResultSet rs=pst.executeQuery();
System.out.println("ENO ENAME ESAL EADDR");
System.out.println("-----");
while(rs.next()){
    System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getFloat(3)+" "
"+rs.getString(4));
}
} z
}
}

```

## JdbcApp29:

```

package com.durgasoft;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Properties;

public class JdbcApp29 {
    public static void main(String[] args) throws Exception{
        Class.forName("com.mysql.jdbc.Driver");
        Properties p=new Properties();
        p.setProperty("user", "root");
        p.setProperty("password", "root");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/durgadb",p);
        PreparedStatement pst=con.prepareStatement("insert into student values(?,?,?,?,?)");
        pst.setString(1, "S-101");
        pst.setString(2, "Durga");
        pst.setString(3, "Java");
        Date d=Date.valueOf("2015-01-25");
        pst.setDate(4, d);
        pst.executeUpdate();

        pst.setString(1, "S-222");
        pst.setString(2, "Anil");
        pst.setString(3, "Oracle");
        java.util.Date date=new java.util.Date();
        int val=date.getDate(); long l = date.getTime();
        java.sql.Date dt=new java.sql.Date(val); Date date2 = new Date(l);
        pst.setDate(4, dt); Date2
        pst.executeUpdate();
    }
}

```

*S.0.println("Employee records inserted successfully");*

*PreparedStatement ps1=con.prepareStatement();*

*ResultSet rs=ps1.executeQuery();*

*S.0.println("ENO "+ "DOJ");*

*S.0.println(" "+ " ");*



14-10-2015

Ex:- (24) →

Ex:- (25) →

Ex:- (26) →

Ex:- (27) →

Ex:- (28) →

Ex:- (29) \* Insert the <sup>date</sup> data and retrieve the date

mysql> create table emp2(eno int(5), primary key, DOJ date);  
 > commit; (first create emp2 table with this instruction in  
 mysql db then perform the following program).

```
import java.sql.*;
public class TableApp29{
  public static void main(String[] args) throws Exception {
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/
    clergadb", "root", "root");
    PreparedStatement pst = con.prepareStatement("insert into emp2
    value(???)");
  }
}
```

To insert the  
date to PST.setInt(1, 111);

```
Date date1 = Date.valueOf("2015-12-12");
pst.setDate(2, date1);
pst.executeUpdate();
```

PST.SetInt(1, 222);

To insert the  
current date
 <sup>java.util.Date</sup> date = new java.util.Date();
 System.out.println(date);
 long l = date.getTime();

```
Date date2 = new Date(l);
pst.setDate(2, date2);
pst.executeUpdate();
```

S. O. P("Employee Records Inserted Successfully");

// To retrieve the data from DB Table.
 PreparedStatement PST1 = con.prepareStatement("Select \* from
 emp2");

ResultSet rs = PST1.executeQuery();

```

S.O.P17("ENO1+DOJ"),
S.O.P17("-----");
while(rs.year()) {
S.O.P17(rs.getInt(1) + "\t" + rs.getDate(2));
}
pst.close();
con.close();
}
}

```

→ Here, to retrieve the date data from emp2, first we have to delete the emp2 then retrieve the data, because we have used primary key which does not allow duplication.

mysq> Create table emp2(eno int(5), primary key, doj date);  
> Commit;

Note:- In false applications, dates manipulations like inserting and retrieving dates is possible with Only prepared statement, not with Statement.

If we want to get a particular type of Resultset object through prepared statement then we have to use the following method.

~~PreparedStatement PSP = con.prepareStatement("")~~

```

public PreparedStatement prepareStatement(String queryFmt,
    int forwardOnly/ScrollSensitive/ScrollInsensitive, int
    Read-only/Updatable) throws SQLException

```

Ex:-

```

PreparedStatement PST = con.prepareStatement("Select
* from emp1", ResultSet.TYPE_SCROLL_SENSITIVE,

```

15/10/2015

(59)

ResultSet`.CONCUR_UPDATABLE`;

ResultSet`rs = ps.executeQuery();`

NetBeans IDE :-

15/10/2015

Home: Sun Microsystems (Oracle Corp)

Objective: To simplify JAVA/J2EE application development.

S/W Type: Open Source Software

Product Type: IDE (Integrated Development Environment)

First Version: NetBeans ~~3.5~~ 3.5 (2003, JUN)

Used Version: NetBeans 7.4 (2013, Oct)

Latest Version: NetBeans 8.0, 8.0.1, 8.0.2

Steps to Design the JDBC Application in NetBeans IDE:-

① Create Java application:-

File → New project → java → java application → Next

project name: JDBCApp

Click on "Finish" button

② Add Driver jar file to Project Library:-

Right click on Libraries under project → select "Add Jar/Folder"

→ Select OJDBC6.jar file → Click on Open button.

(Note:- Write JDBC Application logic in Main class, in main() method.)

### ③ Run JDBC Application:-

Right click on Project → Select "Run".

### Batch Updates:-

Ex:- 30 → page no-62.

Ex:- 31 → page no-62.

### Connection Pooling:-

In general, in JDBC applications, we will create Connection Object when we want to interact with database in order to perform database operations and we will destroy Connection Objects when database operations are completed.

If we use above approach in JDBC applications, then performance of JDBC application is reduced, because creating Connection Object and destroying Connection Objects are two strength full operations.

In the above context, to improve application performance we have to use "Connection Pooling".

In connection pooling, we will create a pool object with no. of connection objects at the time of loading our application (At the time of server startup), we will get connection object from pool object when we want to perform database operations, after performing database operations we have to send Connection Object back to pool object without destroying object.

If we want to implement connection pooling in JDBC applications we have to use the following steps:-

## ① Create Datasource Object:-

Datasource is an object, it able to manage all the JDBC parameters which are required to create connection objects like Driver URL, database username, database password.

To represent datasource, JDBC has provided a predefined interface in the form of `javax.sql.DataSource`, but its implementation classes are provided by all the db vendors.

Oracle has provided the following implementation classes for Datasource interface in order to implement connection pooling.

`Oracle.jdbc.pool.OracleDataSource`

`Oracle.jdbc.pool.OracleConnectionPoolDataSource`

Ex:- `OracleDataSource ds = new OracleDataSource();`

E

## ② Set JDBC Parameters:-

To set JDBC parameters to Datasource Object, we have to use the following methods.

`public void setURL(String driver_URL)`

`public void setUsername(String db-User-Name)`

`public void setPassword(String db-Pwd)`

Ex:-

`ds.setURL("jdb:oracle:thin:@localhost:1521:xe");`

`ds.setUsername("System");`

`ds.setPassword("durga");`

S

### (3) Get Connection Object from Datasource :-

To get Connection object from Datasource, we have to use the following method.

public Connection getConnection() throws SQLException.

Ex:- Connection con = ds.getConnection();

Ex:- f48

```
import java.sql.*;
import Oracle.jdbc.pool.OracleDataSource;
public class JdbcApp48 {
    public void main(String[] args) throws Exception {
        OracleDataSource ds = new OracleDataSource();
        ds.setURL("jdbc:oracle:thin:@localhost:1521:xe");
        ds.setUser("system");
        ds.setPassword("Durga");
        Connection con = ds.getConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("Select * from emp1");
        System.out.println("ENO" + "TENAME" + "ESAL" + "EADDR");
        System.out.println("-----");
        while (rs.next()) {
            System.out.print(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getFloat(3) + " " + rs.getString(4));
        }
        st.close();
        con.close();
    }
}
```

*it is an interface provided by Sun Microsystems*

✓ JdbcApp30:

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

public class JdbcApp30 {
    public static void main(String[] args) throws Exception {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
            "system", "durga");
        Statement st = con.createStatement();
        st.addBatch("insert into emp1 values(666,'FFF',9000,'Hyd')");
        st.addBatch("update emp1 set esal=esal-500 where esal<10000");
        st.addBatch("delete from emp1 where eno=555");
        // st.addBatch("select * from emp1");--> java.sql.BatchUpdateException
        int[] rowCounts = st.executeBatch();
        for (int i = 0; i < rowCounts.length; i++) {
            System.out.println("Records Manipulated : " + rowCounts[i]);
        }
        st.close();
        con.close();
    }
}
```

## Batch Updations with PreparedStatement:

DURGASOFT

```
JdbcApp31:
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class JdbcApp31 {
    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/durgadb",
            "root", "root");
        PreparedStatement pst = con.prepareStatement("insert into emp1 values(?, ?, ?, ?)");
        pst.setInt(1, 666);
        pst.setString(2, "FFF");
        pst.setFloat(3, 6000);
        pst.setString(4, "Hyd");
        pst.addBatch();
    }
}
```

```

pst.setInt(1,777);
pst.setString(2, "GGG");
pst.setFloat(3, 7000);
pst.setString(4, "Hyd");
pst.addBatch();

pst.setInt(1,888);
pst.setString(2, "HHH");
pst.setFloat(3, 8000);
pst.setString(4, "Hyd");
pst.addBatch();
int[] rowCounts=pst.executeBatch();
for(int i=0;i<rowCounts.length;i++){
    System.out.println("Records Manipulated : "+rowCounts[i]);
}
con.close();
}
}

```

*for (int rowCount : rowCounts) {*

Note: If we include selection group SQL query in a batch then JVM will raise an Exception like  
`java.sql.BatchUpdateException: invalid batch command: invalid SELECT batch command.`

## Transaction Management:-

### JTA (Java Transaction API)

**Transaction:** Transaction is an unit of work performed by the front end application on back end system.

1. Deposit some amount in an account.
2. Withdraw some amount from an account.
3. Transfer some amount from one account to another account.

In database applications, every transaction must satisfy the following 4 properties.

1. Atomicity
2. Consistency
3. Isolation
4. Durability

→ These are also called ACID property.

#### 1. Atomicity:-

In general we are able to perform multiple number of operations in a particular transaction, where performing all the operations or performing none of the operations is called as Atomicity property. If we perform all the operations successfully in a transaction then the state of the transaction should be success.

## 1 Step of creating Datasource Object for MySQL DB:-

(6)

Ex(48)

→ For `[javax.sql.DataSource]` interface provided by SUN Microsystems, MySQL Database has provided an implementation class in the form of `[com.mysql.jdbc.jdbc2.optional.MysqlDataSource];`

Ex:(49)

```
import com.mysql.jdbc.jdbc2.  
import java.sql.*;  
public class JdbcApp49 {  
    public void main(String[] args) throws Exception {  
        MysqlDataSource ds = new MysqlDataSource();  
        ds.setURL("jdbc:mysql://localhost:3306/dergadb");  
        ds.setUser("root");  
        ds.setPassword("root");  
        Connection con = ds.getConnection();  
        Statement st = con.createStatement();  
        ResultSet rs = st.executeQuery("Select * from emp1");  
        System.out.println("ENO\tENAME\tESAL\tEADDR");  
        System.out.println("-----");  
        while(rs.next()) {  
            System.out.println(rs.getInt(1) + "\t" + rs.getString(2) + "\t" + rs.getFloat(3) + "\t" +  
                rs.getString(4));  
        }  
        st.close();  
        con.close();  
    }  
}
```

11

In JDBC applications, we are able to implement connection pooling by using

XXXConnectionPool/DataSource  
XXXConnectionPool/DataSource  
javax.sql.PooledConnection

classes also. If we use these we have to get object we have to get Connection object.

→ To get PooledConnect Object explicitly, we have to use the following method.

public PooledConnection get PooledConnection();

To get connection from PooledConnection we have to use the following method.

~~public Connection get Conn;~~

public Connection getConnection() throws SQLException.

Ex:- (56)

import com.mysql.jdbc.jdbc2.optional.

import java.sql.\*;

import javax.sql.PooledConnection;

public class JDBCApp50 {

    public void main(String[] args) throws Exception {

        MysqlConnectionPoolDataSource ds = new MysqlConnectionPoolDataSource();

        ds.setURL("jdbc:mysql://localhost:3306/durgsoft");

        ds.setUser("root");

        ds.setPassword("root");

        PooledConnection pc = ds.getPooledConnection();

```

Connection Con = pg.getConnection();
Statement St = Con.createStatement();
ResultSet rs = St.executeQuery ("select * from emp1");
S.O.P1n ("ENOLTENAME|t ESAL|t EADDR");
S.O.P1n ("-----");
while(rs.next());
S.O.P1n (rs.getInt(1) + " |t " + rs.getString(2) + " |t " + rs.getFloat(3) + " |t " +
          rs.getString(4));
}
St.close();
Con.close();
}

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

27/10/2015

### Callable Statement:-

In database applications, if the application requirement is simple then it is suggestable to use SQL queries. If the app requirement is complex then it is not suggestable to use SQL queries directly, where it is suggestable to use stored procedures and functions.

Q: What is the difference between Stored procedures and functions?

→ Stored procedure is a set of SQL queries managed at database, representing a particular action and it will not return any value by using "return" statement.

#### Syntax:-

Create or replace procedure proc-name [(Param-List)]

As

----- Global Variables  
----- ~~Database logic~~ -----

BEGIN

----- Database logic -----

END Proc-Name;

/--> To save and compile procedure.

Stored function is a set of SQL queries managed at database, representing a particular action and it will return a value by using "return" statement.

#### SYNTAX:-

Create or replace function fun-name [(Param-List)] return

Data-Type

As

----- Global Variables -----

BEGIN

----- Database logic -----

return value;  
END FUN-Name;

/ ----> To save and compile function.

There are three types of parameters to the procedures and functions.  
IN Type Parameter:-

→ It will get value from procedure call to procedure body.

Ex:- e10 IN number

Out Type Parameter:-

→ It will get value from procedure body to procedure call.

Ex:- esal OUT Number.

INOUT Type Parameter:-

→ It is acting as both IN type parameter and Out type parameter.

Ex:- e10 INOUT number.

≡

\* If we want to access stored procedures and functions available at Database from java applications then we have to use java.sql.CallableStatement.

If we want to use CallableStatement in JDBC applications then we have to use the following steps.

(1) Create CallableStatement Object:-

To Create CallableStatement object we have to use the following method from java.sql.Connection.

public CallableStatement prepareCall(String proc-call)

Ex:- CallableStatement cst = con.prepareCall("{call getSal(?, ?)}");

When JVM encounters the above instruction, JVM will take procedure call, JVM will send that procedure call to Database engine, where Database engine will perform Query Tokenization, Query Parsing, Query Optimization and Query Execution then Database engine will create a buffer with the positional parameters called as query plan or Execution plan. w.r.t. the query plan, JVM will create CallableStatement object with the same positional parameters at java application.

### ② Set Values to IN type parameters:-

→ To set values to IN Type parameters we have to use the following methods.

public void setXXX(int param\_Index, xxx value) throws SQLException  
where XXX may be byte, short, int, ...

Ex:- cst.setInt(1, 111);

### ③ Register Out Type parameters:-

To register Out type parameters we have to use the following method.

public void registerOutParameter(int param\_index, int type)  
Where type may be the constants like BYTE, SHORT, INTEGER, FLOAT,.. from java.sql.Types class.

Ex:- cst.registerOutparameter

S

## (4) Execute Procedure:-

(64)

To execute procedure we

public void execute() throws SQLException

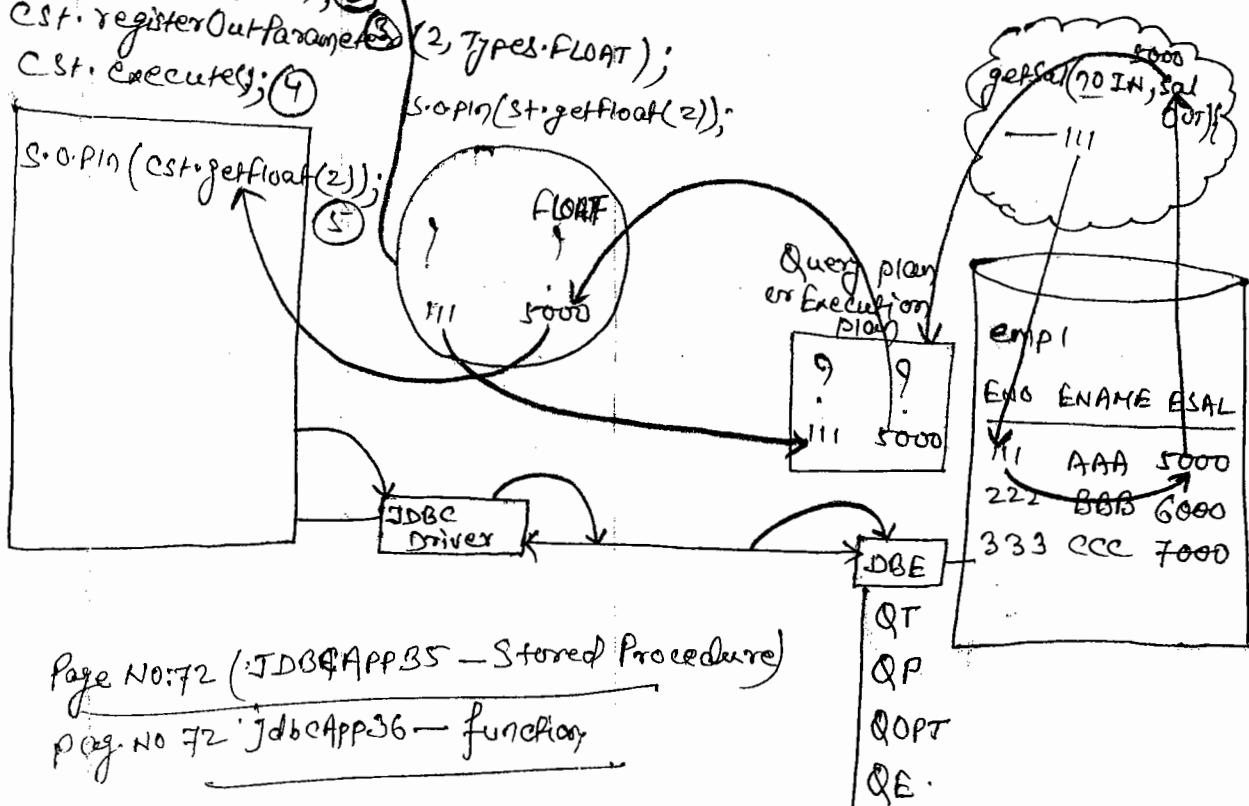
Ex:- Cst.execute();

## (5) Get values from OUT Type parameter:-

To get value from OUT Type parameter we have

public xxx getxxx(<sup>1</sup>int param - Index) throws SQLException  
where xxx may be byte, short, int, ...  
Ex:- float esal = cst.getfloat(2);

CallableStatement cst = con.prepareCall("{call getsal(??)}"); ①  
cst.setInt(1, 111); ②  
cst.registerOutParameter(2, Types.FLOAT); ③  
cst.executeUpdate(); ④  
S.OPIN(cst.getFloat(2));



Page No: 72 (JDBCAAPP35 - Stored Procedure)

Page No 72 JDBCAAPP36 - function

QT  
QP  
QOPT  
QE.

In stored procedures and functions, to represent single value we are able to use the data types like number, float, varchar, ... As per the requirement, if we want to return all the records of a particular table then, first we have to represent all the records data, for this we have to use CURSOR types. To represent records of data in stored procedures and functions Oracle database has provided a predefined CURSOR type that is SYS-REFCURSOR. If we want to use the cursor types in applications then we have to use the following steps.

① IN Stored procedures and functions

a) Declare an OUT type parameter as SYS-REFCURSOR type:

EMP OUT SYS-REFCURSOR

b) Open CURSOR type variable before SQL query execution.

Open emp for

Select \* from emp;

② In JDBC Applications

a) Register CURSOR Type parameter with OracleTypes.CURSOR

Cst.registerOutParameter(2, OracleTypes.CURSOR);

Here, OracleTypes class is provided by Oracle in OJDBC6.jar file in the form of oracle.jdbc.OracleTypes.

b) Get Records of data in the form of Resultset object from CURSOR Type parameter.

public Object getObject(int param\_index)

Ex:- ResultSet rs = (ResultSet)cst.getObject(2);

Ex:- page → 33

JdbcApp → 37

JdbcApp37:

```

/*
create or replace procedure getEmps(sal IN number, emps OUT SYS_REFCURSOR)
AS
BEGIN
    open emps for → we are using this bcz of
    select * from emp1 where esal < sal →
    AS emp sys-REFCURSOR;
    BEGIN
        open emps for
        select * from emp1 where
        esal < sal →
END getEmps;
/
*/
package com.durgasoft;

```

```

import java.io.FileInputStream;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.Properties;

```

```

import oracle.jdbc.internal.OracleTypes;

public class JdbcApp37 {
    public static void main(String[] args) throws Exception {
        FileInputStream fis=new FileInputStream("db.properties");
        Properties p=new Properties();
        p.load(fis);
        String driver_class=p.getProperty("driver_class");
        String driver_url=p.getProperty("driver_url");
        Class.forName(driver_class);
        Connection con=DriverManager.getConnection(driver_url,p);
        Callablestatement cst=con.prepareCall("{call getEmps(?,?)}");
        cst.setFloat(1, 10000);
        cst.registerOutParameter(2, OracleTypes.CURSOR);
        cst.execute();
        Object obj=cst.getObject(2);
        ResultSet rs=(ResultSet)obj;
        System.out.println("ENO\tENAME\tESAL\tEADDR");
        System.out.println("-----");
        while(rs.next()){
            System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+"\t"+rs.getFloat(3)+"\t"+rs.getString(4));
        }
        con.close();
    }
}

```

SQL - Range

AS emp sys-REFCURSOR;

BEGIN

open emps for

select \* from emp1 where

esal < sal;

① Class.forName("oracle.jdbc.OracleDriver");

② Connection con = DriverManager.getConncetion(" ");

③ Connection con = DriverManager.getConncetion(" ");

④ Callablestatement cst=con.prepareCall("{call getEmps(?,?)}");

cst.setFloat(1, 10000);

cst.registerOutParameter(2, OracleTypes.CURSOR);

cst.execute();

Object obj=cst.getObject(2);

ResultSet rs=(ResultSet)obj;

System.out.println("ENO\tENAME\tESAL\tEADDR");

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

while(rs.next()){

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

## db.properties

```
driver_class=oracle.jdbc.OracleDriver
driver_url=jdbc:oracle:thin:@localhost:1521:xe
user=system
password=durga
```

## JdbcApp38:

```
/*
create or replace function getEmployees(no1 IN number,no2 IN number) return SYS_REFCURSOR
AS
employees SYS_REFCURSOR;
BEGIN
open employees for
    select * from emp1 where eno>=no1 and eno<=no2;
return employees;
END getEmployees;
/
*/
package com.durgasoft;

import java.io.FileInputStream;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.util.Properties;

import oracle.jdbc.internal.OracleTypes;

public class JdbcApp38 {
    public static void main(String[] args) throws Exception {
        FileInputStream fis=new FileInputStream("db.properties");
        Properties p=new Properties();
        p.load(fis);
        String driver_class=p.getProperty("driver_class");
        String driver_url=p.getProperty("driver_url");
        Class.forName(driver_class);
        Connection con=DriverManager.getConnection(driver_url, p);
        CallableStatement cst=con.prepareCall("{?call getEmployees(?,?)}");
        cst.setInt(2, 111);
        cst.setInt(3, 555);
        cst.registerOutParameter(1, OracleTypes.CURSOR);
        cst.execute();
        ResultSet rs=(ResultSet)cst.getObject(1);
        cst.getfloat(2, 10000.0f);
    }
}
```

*sal range IN number*

*eno >= sal-range*

*eno <= sal-range*

*Connection con = -*

*for sal range*

*To catch the return value*

```
System.out.println("ENO\tENAME\tESAL\tEADDR");
System.out.println("-----");
while(rs.next()){
    System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+"\t"+rs.getFloat(3)+"\t"+rs.getString(4));
}
}est.close();
con.close();}
```

#### db.properties

```
driver_class=oracle.jdbc.OracleDriver
driver_url=jdbc:oracle:thin:@localhost:1521:xe
user=system
password=durga
```

## ~~Connection Pooling:~~

In general in Jdbc applications, when we have a requirement to perform database operations we will establish the connection with the database from a Java application, at the end of the application we will close the connection i.e. destroying Connection object.

In Jdbc applications, every time establishing the connection and closing the connection may increase burden to the Jdbc application, it will reduce the performance of the jdbc application.

In the above context, to improve the performance of Jdbc applications we will use an alternative called as Connection Pooling.

In Connection pooling at the time of application startup we will prepare a fixed number of Connection objects and we will keep them in a separate base object called Pool object.

In Jdbc applications, when we have a requirement to interact with the database then we will get the Connection object from Pool object and we will assign it to the respective client application.

At the end of the Jdbc application we will keep the same Connection object in the respective Pool object without destroying.

The above mechanism will improve the performance of the application is called as Connection Pooling.

If we want to implement Connection pooling in Jdbc application we have to use the following steps.

**Step 1: Prepare DataSource object.**

DataSource is an object, it is able to manage all the Jdbc parameter which are required to establish the connections.



Ex:- TableApp38

Page no: 74

30/10/2015

## Transaction Management :-

Transaction is an unit of work performed by front end application or Back end System.

Ex:- Performing the Operations like deposit, withdraw, transfer amount from one account to another account in Bank applications are treated as transactions.

In database applications, all the transactions must follow the following properties called as ACID properties.

### (1) Atomicity :-

In transactions, we may perform one or more no. of operations as per the requirement, here "perform either all the operations or none of the operations" is called as atomicity property.

### (2) Consistency :-

In transactions, before executing the transaction and after executing the transaction, the state of the database must be stable.

### (3) Isolation :-

If we execute more than one transaction on a single data item then that transactions are called as concurrent transactions, here, one transaction execution should not give effect to another transaction execution, this property is called as Isolation.

### (4) Durability :-

In Database applications, after committing a transaction if we have any system failures and after getting back the system all the modifications which we performed during the transaction must be preserved, this property is called as Durability.

If we want to provide transactions in JDBC applications, then JDBC application must implement all the above ACID properties.

In JDBC applications, When we establish connection between java application and database then connection is having a default nature that is auto-commit.

In case of Connection Auto-Commit nature, if we submit an SQL query to connection then connection will carry that SQL query to database Engine and connection will make database Engine to execute SQL query and to store results onto the database permanently.

In JDBC applications, Connections Auto-Commit Nature is violating Automicity property of the transactions.

If we want to prevent Automicity property of the transactions, We have to remove Auto-Commit nature of the transactions and we have to perform either "Commit" or "Rollback" Operations at the end of the transactions.

To remove Auto-Commit nature of the transactions, we have to use the following method.

```
public void setAutoCommit(boolean b)
```

→ If b value is true then Connection is in Auto-Commit.

→ If b value is false then Connection is not in Auto-commit.

To <sup>perform</sup> Commit or Rollback Operations explicitly the method we use is:-

```
public void commit() throws SQLException
```

```
public void rollback()
```

SNO	SNAME	SMARKS
111	AAA	78
222	BBB	89
333	CCC	66

try {

```

    Con.setAutoCommit(false);
    Set.ELU("insert --- Student(111, 'AAA', 78 )");
    Set.ELU("insert ----- Student(222, 'BBB', 89 )");
    Set.ELU("insert - - - Student(333, 'CCC', 66 )");
  
```

Con.commit();

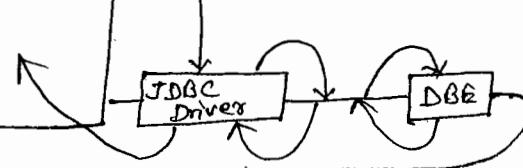
} catch (Exception e) {

Con.rollback();

e.printStackTrace();

}

Con.setAutoCommit(false);

~~Auto-Commit~~

Student

SNO	SNAME	SMARKS
111	AAA	78
222	BBB	89
333	CCC	66

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

Ex:- JdbcApp32 : page 65

Ex:- JdbcApp33 → Page 66

### Savepoint :-

The main intention of savepoint is to store a set of SQL queries execution in transaction commit operation.

To create savepoint we have to use the following method from Connection.

```
public void setSavepoint()
```

To perform rollback operation over the SQL queries which are available from savepoint we have to use the following method.

```
public rollback(Savepoint sp)
```

To remove savepoint we have to use the following method.

```
public void releaseSavepoint(Savepoint sp)
```

(Note:- Savepoint feature is provided by JDBC3.0 version.)

(Note:- Savepoint feature is not supported by Type-1 driver provided by Sun Microsystems, it is supported by Type-4 driver provided by Oracle. Even Type-4 driver is supporting Savepoint feature upto setSavepoint() and rollback(SP) methods only, it is not supporting releaseSavepoint(-) method.)



## JdbcApp33:

```

package com.durgasoft;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class JdbcApp33 {

    public static void main(String[] args) {
        Connection oracle_conn=null;
        Statement oracle_st=null;
        Connection mysql_conn=null;
        Statement mysql_st=null;
        try {
            Class.forName("oracle.jdbc.OracleDriver"); → for oracle
            Class.forName("com.mysql.jdbc.Driver"); → for mysql
            oracle_conn=DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","durga");
            mysql_conn=DriverManager.getConnection
                ("jdbc:mysql://localhost:3306/durgadb","root","root");

            oracle_conn.setAutoCommit(false);
            mysql_conn.setAutoCommit(false);

            Statement oracle_st=oracle_conn.createStatement();
            Statement mysql_st=mysql_conn.createStatement();

            BufferedReader br=new BufferedReader
                (new InputStreamReader(System.in));
            System.out.print("Source Account :");
            String source_Account=br.readLine();
            System.out.print("Target Account :");
            String target_Account=br.readLine();
            System.out.print("Amount To Transfer :");
            int trans_Amt=Integer.parseInt(br.readLine());

            // int oracle_rowCount=oracle_st.executeUpdate
            ("update account set balance=balance-"+trans_Amt+" where
             accNo='"+source_Account+"'");
        }
    }
}

```

After + sign provide  
a Space and  
(double inverted 66  
comma with  
file)

```
// int mysqlRowCount=mysql_st.executeUpdate("update account set  
// balance=balance+" +trans_Amt+ " where accNo='"+target_Account+"'");  
// if((oracle_rowCount==1) && (mysqlRowCount==1)){  
//     oracle_conn.commit();  
//     mysql_conn.commit();  
//     System.out.println(trans_Amt+" Transferred Successfully from  
// "+source_Account+" to "+target_Account); //  
//     System.out.println("Transaction Success");  
// }else{  
//     oracle_conn.rollback();  
//     mysql_conn.rollback();  
//     System.out.println("Transaction Failure");  
}  
}  
} catch (Exception e) {  
    try {  
        oracle_conn.rollback();  
        mysql_conn.rollback();  
        System.out.println("Transantion Failure");  
    } catch (Exception e2) {  
        e2.printStackTrace()  
    }  
}
```

~~CASOFT~~

↑ catch (Exception e)  
(Incomplete after this)

### ~~Java.sql.SavePoint~~

One of the features introduced in JDBC 3.0 version. And it is an interface Savepoint is an intermediate point and makes it possible to rollback the transaction upto the save point instead of rolling back the entire transaction.

To represent Savepoint Jdbc has provided a predefined interface `java.sql.Savepoint`. To set a Savepoint we have to use the following method form Connection.

```
public Savepoint setSavepoint()
```

To perform rollback operation on set of instructions executive w.r.t a particular Savepoint we have to use the following method from Connection.

```
public void rollback(Savepoint sp)
```

To release Savepoint we will use the following method form Connection.

```
public void releaseSavepoint(Savepoint sp)
```



## Transaction Isolation:-

In database applications, if we allow more than one transaction to execute on a single data item then that transactions are called as Concurrent transactions and this process is called as Transaction Concurrency.

In transaction Concurrency, when we execute more than one transaction on a single data item then there may be a chance to get Concurrency problems, which may provide data inconsistency.

In Database applications, Transactions Concurrency may provide the following four types of problems.

- ① Lost Update
- ② Dirty Read
- ③ Non-repeatable Read
- ④ Phantom Read.

To resolve all the above Transaction concurrency problems, java.sql.Connection interface has provided isolation levels in the form of the following constants.

- ① public static final int TRANSACTION\_NONE = 0;  
→ It will not resolve any of the concurrency problems, it will represent all the Concurrency problems.
- ② Public static ~~int~~ final int TRANSACTION\_READ\_UNCOMMITTED = 1;  
→ It will resolve lost update problem and it will represent remaining Concurrency problems.
- ③ public static final int TRANSACTION\_READ\_COMMITTED = 2;  
→ It will resolve lost update, dirty read problems and it will represent all remaining.

- (4) public static final int TRANSACTION\_REPEATABLE\_READ = 4;  
 → It will resolve lost update, Dirty Read and non-repeatable problem and it will represent phantom Read problem.
  - (5) public static final int TRANSACTION\_SERIALIZABLE = 8;  
 → It will solve all the Concurrency problems.  
 The default isolation level of Connection Object is  
 $\text{TRANSACTION\_READ\_COMMITTED} = 2;$
- 

→ To set the above specified Isolation levels to Connection we have to use the following method.

public void setTransactionIsolation(int type)

Eg:-

con.setTransactionIsolation(Connection.TRANSACTION\_SERIALIZABLE);

To get Connection's Isolation level we have to use the following method.

public int getTransactionIsolation()

Eg:- int type = con.getTransactionIsolation();

### ① Lost Update:-

- (a) Transaction T1 has performed operations on a particular data item but Commit operation is not performed.
- (b) Transaction T2 has performed operations over the same data and T2 transaction has performed Commit operation.
- (c) After T2 transaction's Commit operation, T1 transaction ~~has~~ performed Commit operation.

- (d) In the above Concurrent Transactions, T1 transaction lost its updations, so that this problem is called as Lost Update.

T1	bal=500	T2
Read(bal);		
bal=bal+500;		
Write(bal);		
Commit();		

Read(bal);	
Bal = Bal + 1000;	
Write(bal);	
Commit();	

To resolve this problem, we have to use the following instruction.

Con.setTransactionIsolation(Connection.TRANSACTION\_READ\_UNCOMMITTED - TED);  
or

Con.setTransactionIsolation(1);

## ② Dirty Read Problem:-

- (a) TRANSACTION T1 has performed updations on a particular data items without performing Commit and rollback operation.
- (b) Transaction T2 has performed read Operation over the Uncommitted data modified by transaction t1.
- (c) After T1 transaction read Operation, T1 transaction has performed rollback Operation over it's updations.
- (d) Due to T1 transaction's rollback operation, T2 transaction read operation is becoming as Dirty Read, so, that this problem is called as Dirty Read Problem.

T1	bal=500	T2
Read(bal); bal=bal+500; Write(bal);		Rollback(Bal); → Dirty Read
Rollback(); Commit();		

To resolve Lost Update and Dirty Read Problems in DBMS apps we have to use the following instructions.

Conn.setTransactionIsolation(Connection.TRANSACTION\_READ\_COMMITTED);  
or

Conn.setTransactionIsolation(2);

### ③ Non-Repeatable Read Problem:-

- ② Transaction T1 is performing a sequence of read operations over a particular data item, every time T1 transaction ~~has~~ has to generate same results as per its requirement.
- ③ Between two read Operations, Transaction T2 is performing updations over the same data item.
- ④ After T2 transaction updations, In T1 transaction read operation same results are not generated, so that this problem is called of Non-Repeatable Read problem.

04/11/2015

(69)

To resolve this problem, we have to use the following instruction.

CON-SET TRANSACTION ISOLATION (Connection.TRANSACTION-REPEATABLE-READ);  
LORI

CON-SET TRANSACTION ISOLATION(4);

	T1	bal=500	T2
A	A	Read();	update();
B	A'	Read();	update();
C	A'	Read();	update();
D	A'	Read();	

#### ④ Phantom Read

04/11/2015

- ① T1 transaction is performing a sequence of read operation and it is expecting same no. of results at each and every read operation.
- ② Between two consecutive read operations of T1 transaction, T2 transaction is performing insert operations.
- ③ After performing insert operations, A T1 transaction is performing read operation then T1 transaction is not getting same no. of results.

To resolve all concurrency problems we have to use the following instruction.

CON-SET TRANSACTION ISOLATION (Connection.TRANSACTION-SERIALIZABLE);

Z

P.T.O

### Results:-

	T1	T2
10	Read()	
15	10 Read()	Insert()
20	10 Read()	Insert()
25	10 Read()	Insert()

### \* BLOB :

This is a data type at databases to store large volumes of binary data.

If we want to represent an image file in databases we have to use BLOB data type.

If we want to store an image data in databases we have to use the following steps.

① Create a table at database with BLOB type column:-

```
SQL> create table emp10 (eno number, emp_image, blob);  
SQL> Commit;
```

② Get Image file into java application in the form of File class Object:-

```
File f = new File("abc.jpg");
```

③ Get Image data from file class object to FileInputStream:-

```
FileInputStream fis = new FileInputStream(f);
```

④ Create PreparedStatement with insert SQL query:-

PreparedStatement pst = con.prepareStatement("Insert into emp10 values(??);");

⑤ Set data to positional parameters in PreparedStatement:-

pst.setInt(1, 111);

To Stream of binary data to positional parameter, we have to use the following method.

public void setBinaryStream (int param\_index, FileInputStream fis, int size)

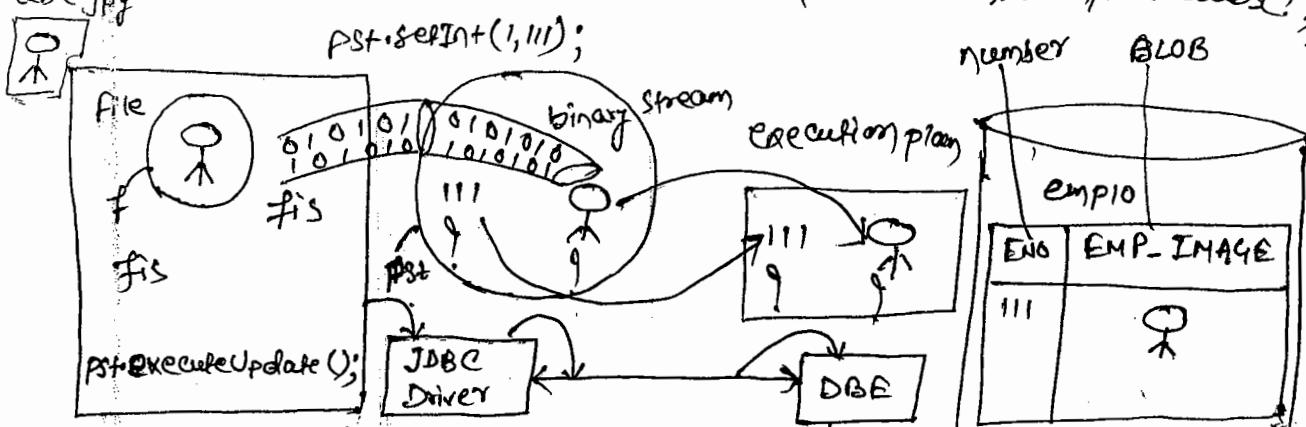
Ex:- pst.setBinaryStream(2, fis, f.length());

⑥ Execute insert SQL query:-

pst.executeUpdate();

PAGE NO → 78, JDBC APP 40

PreparedStatement pst = con.prepareStatement("insert into emp10 values(??);");  
abe.jpg



File f = new File("abe.jpg");

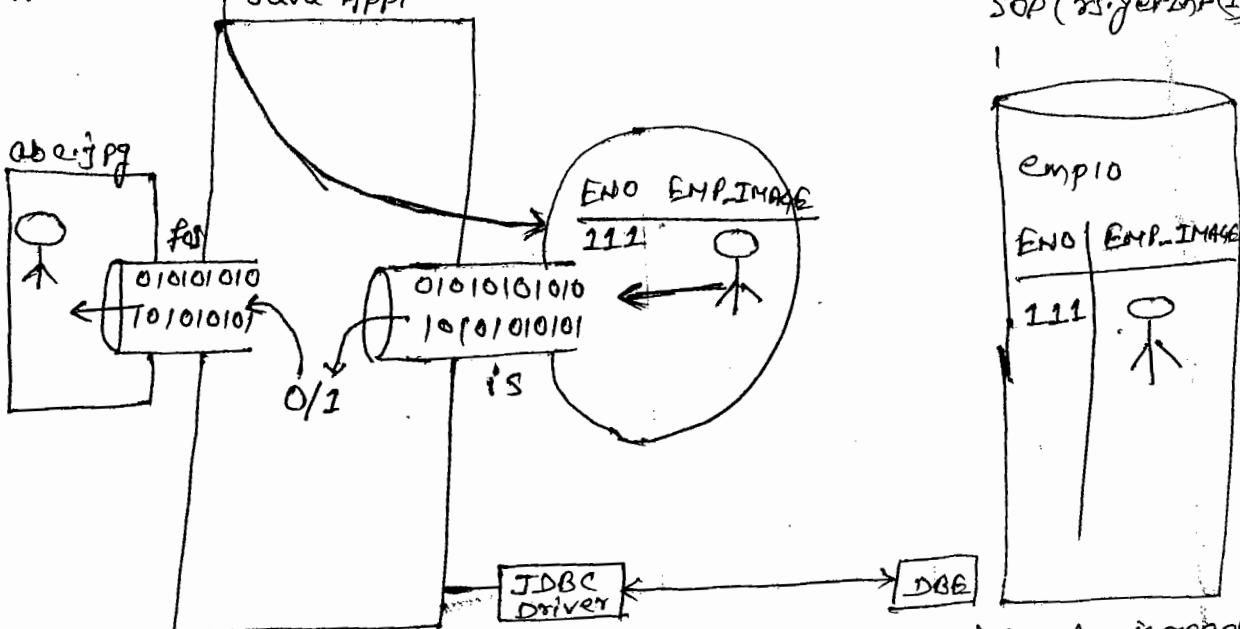
FileInputStream fis =

pst.executeUpdate();

QF  
QP  
QOPT  
QR

## ④ Retrieving of BLOB:-

If we want to retrieve ~~BLOB~~ type of data from database  
 ResultSet rs = st.executeQuery("select \* from emplo");  
 Java Appl



```
FileOutputStream fos = new FileOutputStream("abc.jpg");
InputStream is = rs.getBinaryStream(2);
int val = is.read();
while(val != -1)
    fos.write(val);
```

table sever we have to use the following steps.

### ① Create ResultSet Object:-

```
ResultSet rs = st.executeQuery("select * from emplo");
```

### ② Get blob type of data in the form of InputStream:-

To get BLOB type data from ResultSet Object in the form of InputStream Object then we have to use the following method:

```
public InputStream getBinaryStream(int Column_Index)
```

Ex:-  
`InputStream is = rs.getBinaryStream(2);`

### ③ Prepare Target file to store BLOB type data by using FileOutputStream:-

```
FileOutputStream fos = new FileOutputStream("abc.jpg");
```

(4) Read bit by bit from InputStream and write bit by bit to FileOutputStream:-

(71)

```
    int val = is.read();
    while(val != -1)
    {
        fos.write(val);
        val = is.read();
    }
```

(5) Close all file resources:-

```
is.close();
fos.close();
st.close();
con.close();
```

Ex:-  
Page NO → 79

APP NO → 41

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

CLOB:-

CLOB is a datatype at databases to represent large volumes of character data.

If we want to perform database operations like insertion and retrieve with CLOB data types then we have to use the same conventions of BLOB data type insertion and retrieve operations with the following replacements.

blob → Clob

InputStream → Reader

FileInputStream → FileReader

FileOutputStream → FileWriter

getBinaryStream() → getCharacterStream()

getBinaryStream() → getCharacterStream()

abc.jpg → abc.xml //

Ex:- page-80, JdbcApp42  
page-81, JdbcApp43.

06/11/2015

### RowSet:-

Rowset is an extension of ResultSet object, it able to manage records of data with java bean conventions and having event handling capabilities.

Q. What are the differences between ResultSet and Rowset?

- ⇒ ① ResultSets are not implementing java.io.Serializable interface by default, so that, it is not possible to transfer ResultSet Object through network.

Rowset Objects are implementing java.io.Serializable interface by default, so that, it is possible to transfer Rowset Object through network.

- ② ResultSet Object must required connection with database continuously from java applications.

In Jdbc applications, Rowset Objects are not required connection with database continuously. (Disconnected Rowset).

- ③ Event notification feature is not available in ResultSet Object.

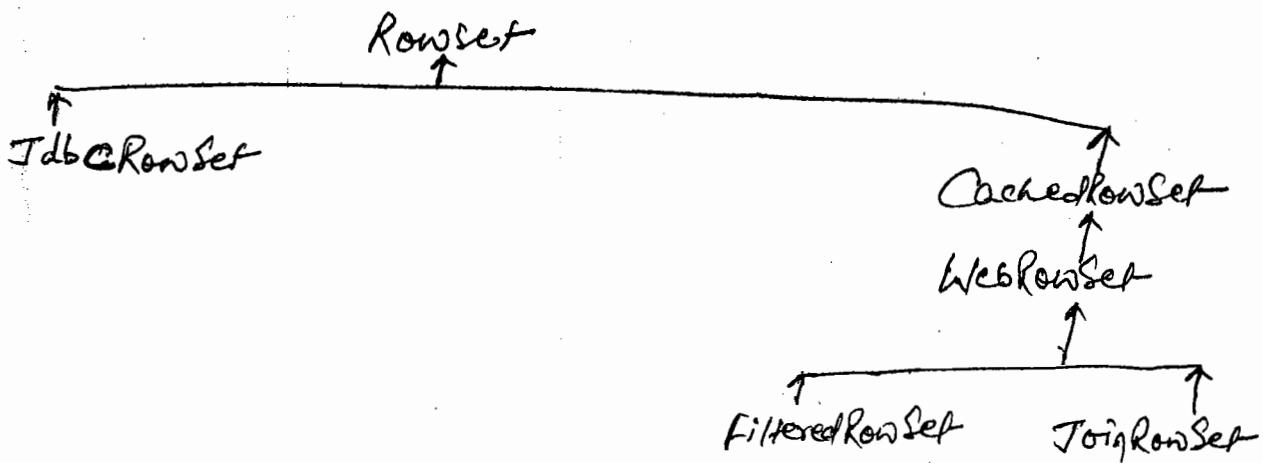
Event notification feature is existed with Rowset Objects.

- ④ By default, ResultSet Object are not Scrollable and Updatable.

By default Rowset Objects are Scrollable and Updatable.

Note:- The main intention of introducing Rowset is  
to simplify the retrieval operations in JDBC application.

JDBC has provided five types of Rowsets in javax.sql.  
Rowset package.



All the above Rowsets are provided by SUN Microsystems in the form of interfaces and their implementation classes are provided by all the database vendors as part of their database softwares.

Note:- SUN Microsystems has provided a reference implementations for all the above Rowsets in the form of the following classes in "com.sun.rowset" package.

- IDbRowsetImpl
- CachedRowsetImpl
- FilteredRowsetImpl
- WebRowsetImpl
- JoinRowsetImpl.

In JDBC, all the above Rowsets are divided into the following two types:-

RR

## ① Connected Rowsets:-

Which must receive connection to the database continuously.

Ex:- ~~Table~~ TableRowset.

## ② Disconnected Rowsets:-

These Rowsets are not required connection to the database continuously, when we want to interact with database then it will establish connection, when we are not interacted with database then it will disconnect the connection.

Ex:-

CachedRowSet-

WebRowSet

FilteredRowSet-

JoinRowSet.

If we want to Rowsets in Table Applications, then we have to use the following steps.

### ① Create Rowset Object :-

TableRowset rs = new TableRowsetImpl();

### ② Set all Table parameters like drive URL, database User Name and database Password to Rowset:-

To perform this action, we have to use the following methods.

public void setURL(String url)

(73)

```
public void setUsername(String username)  
public void setPassword(String password)
```

Ex:-

```
rs.setUrl("jdbc:mysql://localhost:3306/designdb");  
rs.setUsername("root");  
rs.setPassword("root");
```

### (3) Set SQL Query to Rowset:-

To set Select SQL query to Rowset, we have to use the following method.

```
public void setCommand(String sqlQuery);
```

Ex:- rs.setCommand("Select \* from emp");

### (4) Execute SQL query and get results in Rowset Object:-

To execute SQL query and to get results in rowset object we have to use the following method.

```
public void execute();
```

Ex:- rs.execute();

### (5) Retrive Results from Rowset Object:-

To Retrive results from rowset objects, we have to use the same conventions of ResultSet object.

```
while(rs.next())
```

```
    S.O.P1(rs.getInt(1));  
    S.O.P1(rs.getString(2));  
    S.O.P1(rs.getFloat(3));  
    S.O.P1(rs.getString(4));
```



07/11/2015

Ex:-

```
import javax.sql.rowset.JdbcRowSet;
import com.sun.rowset.JdbcRowSetImpl;
public class Jdbcppst {
    public void Psum(String[] args) throws Exception {
        CachedRowSet crs = new JdbcRowSetImpl();
        crs.setUrl("jdbc:mysql://localhost:3306/plurals");
        crs.setUsername("root");
        crs.setPassword("root");
        crs.setCommand("select * from emp1");
        rs.execute();
        System.out.println("EID|ENAME|ESAL|EADDR");
        System.out.println("-----");
        while(rs.next()){
            System.out.println(rs.getInt(1) + " | " + rs.getString(2) + " | " + rs.getFloat(3) + " | " +
                rs.getString(4));
        }
    }
}
```

```

public static void main(String[] args) throws Exception{
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection
    con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","durga"
    );
    Statement st=con.createStatement();
    ResultSet rs=st.executeQuery("select * from employee");
    rs.next();
    System.out.println("Employee Id..... "+rs.getInt(1));
    InputStream is=rs.getBinaryStream(2);
    FileOutputStream fos=new FileOutputStream("myimage.jpg");
    int i=is.read();
    while (i != -1)
    {
        fos.write(i);
        i=is.read();
    }
    // System.out.println("Image created Successfully with the name
    // myimage.jpeg");
    fos.close();
    con.close();
}
}

```

PS+1

## ~~CLOB (Character Large Object)~~

clob is a datatype provided by Oracle, it can be used to represent large values of character data like documents, pdf files and so on.

If we want to perform operations with clob datatype then we have to use the same steps what we have used with blob datatype, but we need to provide the following replacements.

```

JdbcApp42:
import java.sql.*;
import java.io.*;
public class JdbcApp42{
public static void main(String[] args) throws Exception{
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection
    con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","durga"
    );
    PreparedStatement pst=con.prepareStatement("insert into webinfo
    values(?,?)");
    pst.setString(1,"app");
    pst.setString(2,"hibernateapp1");
    File f=new File("web.xml");
}

```

Zia

create table hbn

hibnapps

hibernate.cfg.xml

```
FileReader fr=new FileReader(f);
pst.setCharacterStream(2, fr,(int)f.length());
pst.executeUpdate();
System.out.println("web application stored in database successfully");
pst.close();
con.close();
}
}
```

IdocApp43:

```
import java.sql.*;
import java.io.*;
public class JdbcApp43{
public static void main(String[] args) throws Exception{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","durga");
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select * from webinfo");
rs.next();
System.out.println("Application name is....."+rs.getString(1));
FileWriter fw=new FileWriter("myweb.xml");
Reader r=rs.getCharacterStream(2);
int i=r.read();
while(i != -1)
{
fw.write(i);
i=r.read();
}
//System.out.println("web.xml is retrieved successfully with the name
myweb.xml");
fw.close();
con.close();
}
}
```

## JDBC INTERVIEW QUESTIONS:

1. What is the difference between Database and Database management system?
  2. How a query could be executed when we send a query to Database?
  3. What is Driver? How many Drivers are available in JDBC? What are the types?

The following example shows how to move cursor in various directions like Scrollable Resultset Objects.

Note:- Resultsets are not scrollable by default, but Rowsets are scrollable Bydefault.

Ex:- (5-2)

```

rs.execute();
rs.last();
s.o.pn(rs.getInt(1));
rs.first();
s.o.pn(1);
s.o.pn(rs.getInt(1));
rs.afterLast();
rs.previous();
s.o.pn(rs.getInt(1));
rs.beforeFirst();
rs.next();
s.o.pn(rs.getInt(1));
rs.absolute(3);
s.o.pn(rs.getInt(1));
rs.absolute(-3);
s.o.pn(rs.getInt(1));

```

```

rs.first();
rs.relative(2);
s.o.pn(rs.getInt(1));
rs.last();
rs.relative(-2);
s.o.pn(rs.getInt(1));
s.o.pn(1);

```

The following example shows how to insert records into database table like Updatable Resultset Object.

Eg:

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

```
rs.execute();
```

```
BufferedReader br = new BufferedReader(new InputStreamReader
(system.in));
```

```
rs.moveToInsertRow();
```

```
while(true){
```

```
s.o.p("Employee Number :");
```

```
int eno = Integer.parseInt(br.readLine());
```

```
s.o.p("Employee Name :");
```

```
String ename = br.readLine();
```

```
s.o.p("Employee Salary :");
```

```
float esal = float.parseFloat(br.readLine());
```

```
s.o.p("Employee Address :");
```

```
String eaddr = br.readLine();
```

```
rs.updateInt(1, eno);
```

```
rs.updateString(2, ename);
```

```

rs.UpdateFloat(3, sal);
rs.UpdateString(4, eaddr);
rs.InsertRow();
System.out.println("One more Employee[yes/no]?");
String option = br.readLine();
if(option.equals("No"))
    break;
}

```

The following example shows how to perform updates on database table through Reader Object like Updatable Result-Set Object.

Ex:-

11. executeC()  
while(rs.next())  
float SQL = rs.getFloat(3);

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

```
if (Sal < 10000){  
    Sal = Sal + 500;  
    rs.UpdateFloat(3, Sal);  
    rs.UpdateRow();  
}
```

Note:- In Jdbc application, if you want to delete a particular record from database table through Rowset Object then we have to use the following method from Rowset.

public void ~~Rowset~~ deleteRow();

Ex:- rs.deleteRow();

08/11/2015

## ② CachedRowSet:-

CachedRowSet - It's a disconnected Rowset, it able to manage data in cache memory to improve reusability.

In Jdbc applications, if we want to get a table data frequently then we have to get Rowset object first time and we will keep this Rowset object in Cache memory. In jdbc applications, when we require the same table data then every time we will get table data from Cache memory which we required stored already without interacting with database.

CachedRowSet object will reduce no. of database interactions from java applications, it will improve application performance.

If we want to use CachedRowset in JDBC applications then we have to use the following steps.

- ① Create CachedRowset object.
- ② Set JDBC parameters to CachedRowset like driver URL, database username and password.
- ③ Set select SQL query to CachedRowset.
- ④ Execute SQL query and get data in the form of CachedRowset object.
- ⑤ Get data from CachedRowset object.
- ⑥ If we want to perform insert, update and delete operation on database table through CachedRowset object then we have to use the following methods after insertRow() method, updateRow() method and deleteRow() method.

CRS.moveToCurrentRow();

CRS.AcceptChanges();

The following ~~change~~ example shows how to retrieve data from database table through CachedRowset Object.

Eg:

```
import javax.sql.rowset.CachedRowset;
• CachedRowsetImpl;
```

```

crs.execute();
S.O.Pln("ENO1+ENAME1+ESAL1+EADDR");
S.O.Pln("-----");
while(crs.next()){
    S.O.Pln(crs.getInt(1) + " | " + crs.getString(2) + " | " + crs.getFloat(3)
            + " | " + crs.getString(4));
}
crs.close();
}

```

→ If we want to perform Cursor Scrolling in CachedRowSet object from one position to another position we have to use the TableRowset Conventions. Use TableApp6 application where we have to create CachedRowSet object instead of TableRowset object.

Ex:-

```

import java.io.*;
import javax.sql.rowset.CachedRowSet;
import com.sun.rowset.CachedRowSetImpl;
public class TableApp6{
    PSVM
}

```

```

Crs.execute();
BufferedReader br=new BufferedReader(new InputStreamReader
(Crs.moveToInsertRow());
(System.in));
while(true){
    S.O.P("Employee Number:");
    int eno=Integer.parseInt(br.readLine());
    S.O.P("Employee Name:");
    String ename=br.readLine();
    S.O.P("Employee Salary:");
    float esal=float.parseFloat(br.readLine());
    S.O.P("Employee Address:");
    String eaddr=br.readLine();
    Crs.updateInt(1, eno);
    Crs.updateString(2, ename);
    Crs.updateFloat(3, esal);
    Crs.updateString(4, eaddr);
    Crs.insertRow();
    S.O.P("One more Employee (yes/no)? :");
    String option=br.readLine();
    if(option.equals("no")){
        break;
    }
    Crs.moveToCurrentRow();
    Crs.acceptChanges();
}

```

→ The following example shows how to update database table through CachedRowSet object.

Ex:-

```
import javax.sql.rowset.CachedRowSet;
```

```
crs.execute();
while(crs.next()){
    float esal = crs.getFloat(3);
    if(esal < 10000){
        esal = esal + 500;
        crs.updateFloat(3, esal);
        crs.updateRow();
    }
    crs.moveToCurrentRow();
    crs.acceptChanges();
    crs.close();
}
```

Note:- If we want to delete record from database table through CachedRowSet object then we have to used the following methods.

```
crs.deleteRow();
crs.moveToCurrentRow();
crs.acceptChanges();
```

rec

09/11/2015

(78)

10/11/2015

## WebRowSet:-

In general, in enterprise applications, ~~we are able to transfer the data from~~ applications logic is divided into no. of layers like presentation layer, Business Processing layer, Data Access and Storage layer, ...

Where data Access and Storage layer is able to interact with database in order to perform database operations.

In enterprise applications, it is frequent requirement to carry data from one layer to another layer or from one machine to another machine.

To carry data from one machine to another machine by retrieving data from database we have to use XML document instead of Resultset Object and Rowset Object directly.

In Jdbc applications, if we want to get a particular table data in the form of an XML document and if we want to get data from XML document to Rowset Object then we have to use javax.sql.rowset.WebRowset.

If we want to use WebRowset in Jdbc applications then we have to use the following steps.

① Create WebRowset object:-

```
WebRowset wrs = new WebRowsetImpl();
```

② Set all Jdbc parameters to WebRowset object :-

```
wrs.setURL("jdbc:mysql://localhost:3306/durgadb");  
wrs.setUsername("root");  
wrs.setPassword("root");
```

③ Set "Select" SQL query, execute SQL query and get data from database table to Rowset Object:-

```
wrs.setCommand("Select * from emp1");  
wrs.execute();
```

④ Create FileOutputStream with a particular target XML file:-

```
FileOutputStream fos = new FileOutputStream("emp.xml") ;
```

⑤ Write data from Rowset Object to XML file:-

To write Rowset object data in XML document we have to use the following method.

```
public void writeXML(OutputStream os)
```

Ex:- wrs.writeXML(fos);

Note:- If we want data from XML document to Rowset object then create FileInputStream and use the following method.

```
public void readXML(InputStream is)
```

Ex:-

```
FileInputStream fis = new FileInputStream("emp.xml");  
wrs.readXML(fis);
```

Z

Ex:-

```

import java.io.FileOutputStream;
import javax.sql.rowset.WebRowSet;
import com.sun.rowset.WebRowSetImpl;
public class JDBCApp62{
    public void main(String[] args) throws Exception {
        WebRowSet rs = new WebRowSetImpl();
        rs.setUrl("jdbc:mysql://localhost:3306/dangad6");
        rs.setUsername("root");
        rs.setPassword("root");
        rs.setCommand("Select * from emp1");
        rs.execute();
        FileOutputStream fos = new FileOutputStream("d:/uploads/employee.xml");
        rs.writeXML(fos);
    }
}

```

### Ex:- (XML to Rowset Object)

```

import java.io.FileInputStream;
import javax.sql.rowset.WebRowSet;

```

```
FileInputStream fis = new FileInputStream ("d:/uploads/  
employee.xml");  
rs.readXML(fis);  
S.O.Pn ("ENO|ENAME|ESAL|EADDR");  
S.O.Pn ("-----");  
while (rs.next()) {  
    S.O.Pn (rs.getInt(1) + " | " + rs.getString(2) + " | " + rs.getFloat(3)  
           + " | " + rs.getString(4));  
}
```

~~Remaining FilteredResultSet and JoinResultSet~~

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

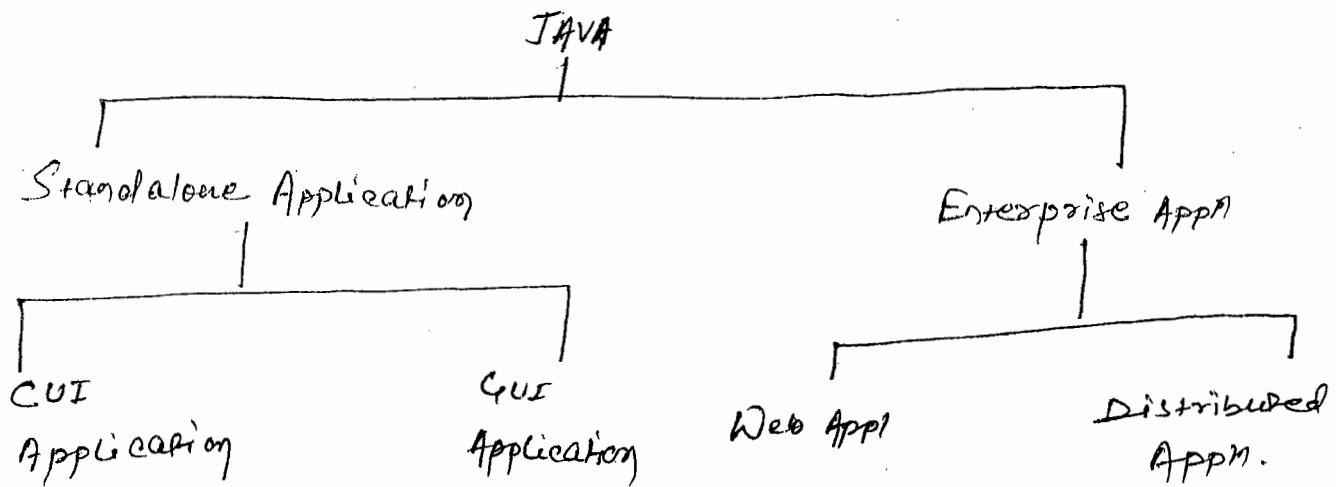
## Servers

12/11/2015

①

### Servlets:

By using java, we are able to design the following two types of applications.

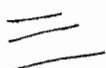


### Standalone Application:-

If we design any java application without using Client-Server architecture then that java application is called as Standalone Application.

There are two types of Standalone Applications.

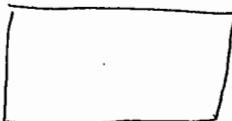
- ① CUI Applications
- ② GUI Application



Q: What is the difference between CUI App and GUI App?

⇒ CUI application is a java application, it would be designed in such a way that to take input through command prompt and to provide out through command prompt, where command prompt is acting as an user interface, it able to support only character data.

Add.java

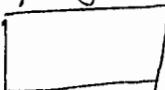


```
D:\app\>java Add  
First Value: 10 I/P  
Second Value: 20 I/P  
Addition : 30 O/P.
```

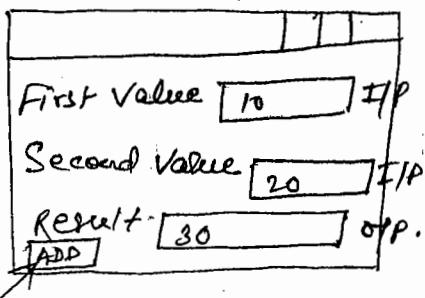
GUI Application is a java app, it would be designed in such a way that to take input through a collection of Graphic Components and to provide output through the same collection of Graphic Components, where Collection of Graphic Components is acting as an user interface; it able to support Graphics data.

(2)

Add.java



D:\apps>java add



## (2) Enterprise Applications:-

If we design any java application on the basis of Client-Server Arch then that java application is called as Enterprise Application.

There are two types of

- (1) Web Application
- (2) Distributed Application.

Q. What are differences between Web Applications and Distributed Applications?

→ (1) Web Application is a client-server appn, where the complete appn logic is distributed over server machine.

Distributed Application is a Client-Server application, where the complete application logic is distributed over Client machine and Server machine.

≡

② In case of web apps, no separate java program is executed at Client machine, at Client machine, browser is acting as Client.

In case of distributed apps, there's a program at Client machine and at Server machine, at Client machine we can use a java program, with main() method, an applet, a servlet, a JSP page, ... acting as Client.

③ To prepare web applications, a set of technologies are used called as web technologies.

Ex:- CGI, Servlets, JSPs, PHP, PERL, .....

To prepare distributed applications a set of technologies are used called as distributed technologies.

Ex:-

Socket Programming

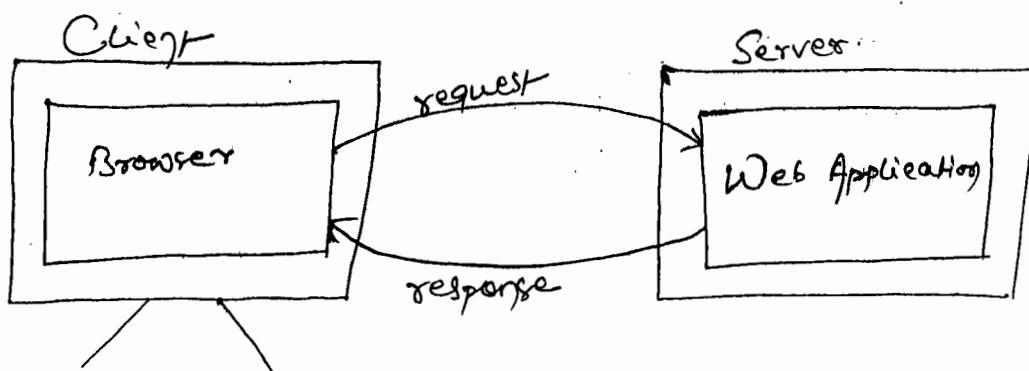
RMI [Remote Method Invocation]

CORBA [Common Object Request Broker Arch/Agent]

EJBs [Enterprise Java Beans]

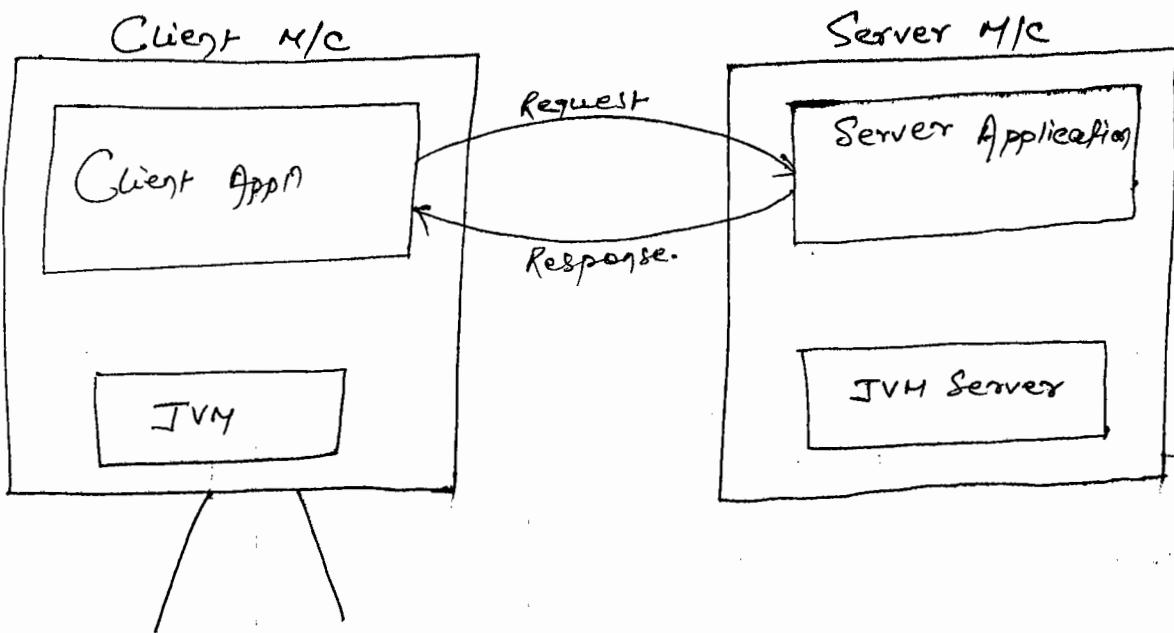
Web Services

Web Applications:-



## ② Distributed Application:-

18/11/2015 ③



④ The main purpose of web applications is to generate dynamic response from Server.

The main purpose of distributed application is to establish connection between local machine and Remote machine in order to get remote services from remote machine.

⑤ Web apps are executed by both web servers and application servers.

Distributed apps are executed by Only application servers.

⑥ Web apps are the collection of web components like Servlets and JSPs, which are executed by web Container.

Distributed applications are the collection of distributed components like EJBs, which are executed by EJB Container.



There are two types of responses in Web applications.

- ① Static response
- ② Dynamic response.

### ① Static Response:-

If we generate any response from server without performing any action or without executing any resource at server machine then that type of response is called as Static Response.

Ex:- Sending requests to an html file available at server and getting html file content as response without performing any action at server.

### ② Dynamic response:-

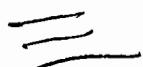
If any response is generated from server by performing an action or by executing a particular resource at server machine then this type of response is called Dynamic Response.

Ex:- Sending a request to Servlet/JSP available at server and generating response to client after executing Servlet/JSP

→ If we want to generate dynamic response from server then we have to execute an app in a server machine, called as Web Application.

To prepare Web Applications, we have to use a set of technologies called as web technologies.

Ex:- CGI, Servlets, JSPs, ....



(4)

Q: To design Web apps we have already CGI then what is the requirement to go for Servlets?

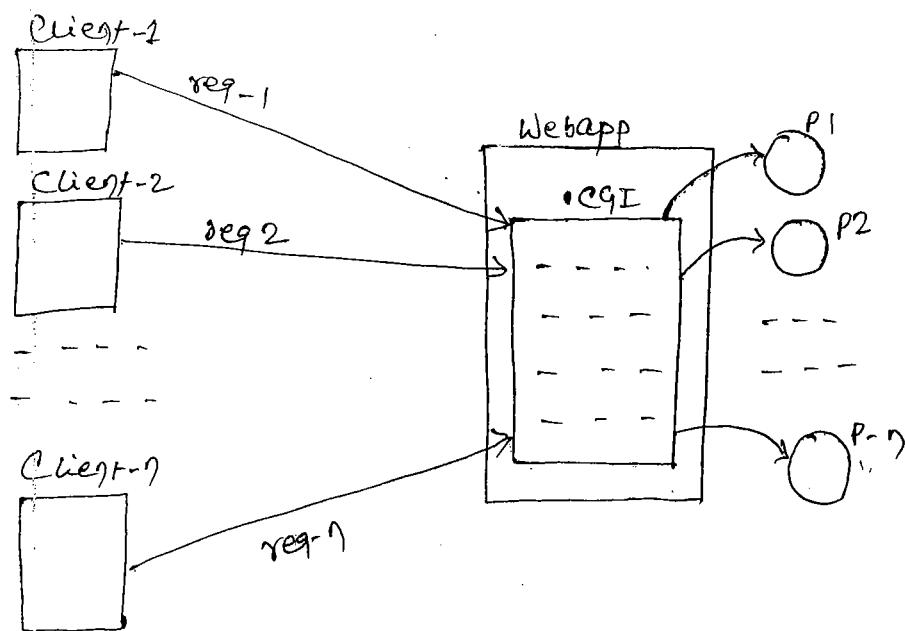
⇒ CGI is a server side technology designed on the basis of C and Scripting language.

'C' is a process based programming language, it will make CGI as a process based server side technology.

If we deploy CGI apps at server then Container will create a separate process for each and every request coming from Client.

Process is a heavy weight component, it will take more execution time and lot of memory to handle, it will increase burden to server machine, it will provide delayed response to clients, It will reduce performance of the server side applications.

In the above context, to improve the performance of server side apps, we have to use thread based technologies to prepare Web applications.



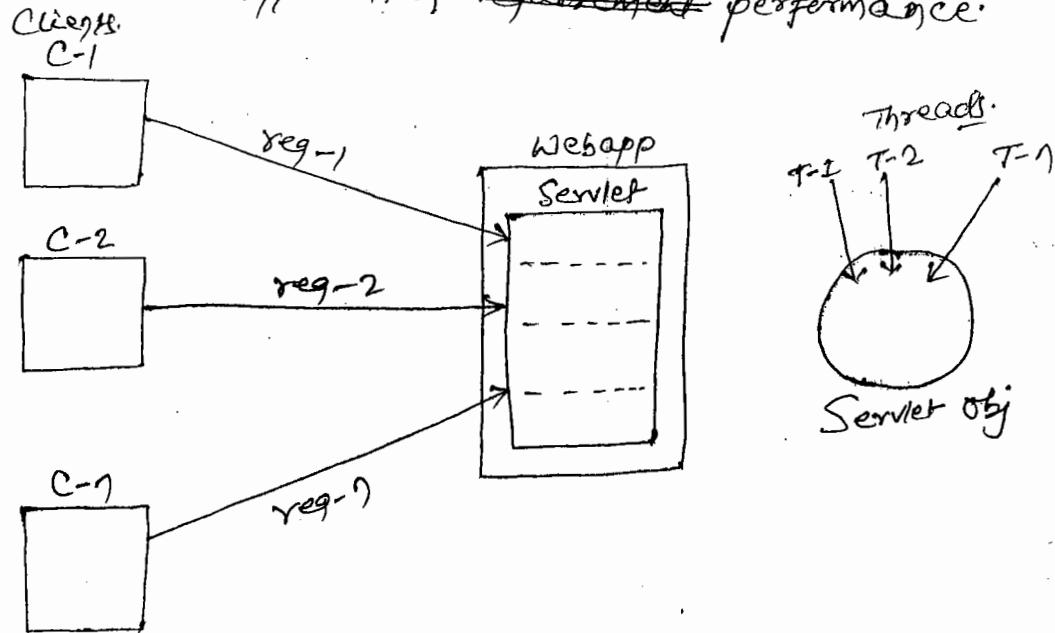
## Servlet Technology:-

Servlet is a server side technology designed on the basis of J2SE.

J2SE [JAVA] is a thread based programming language, it is modelling Servlets is a thread based technology.

If we deploy any Servlet app at server then Container will generate a separate thread for each and every request coming from Client.

Thread is light weight when compared with process, so that Servlets will reduce burden to the Server, it will provide quick responses from Server to Client, it will improve Server Side application ~~requirement~~ performance.



(5)

Q What are the differences between Servlets and JSPs?



1) To design web applications, if we use Servlets then we must require very good awareness on Java Programming.

To design web applications, if we use JSP technology then it is not required to have awareness on java programming.

If we have presentation Servlets and if we know how to use HTML tags then we are able to prepare web applications by using JSP tech.

Note - The main intention of JSP tech is to reduce java code from web applications.

2) In web apps, Servlets are used to pickup the requests from Client and to process the requests.

In web apps, JSPs are used to generate dynamic response to Client with very good look and feel.

3) In web applications, Servlets are used mainly for processing and JSPs are used mainly for presentation purpose.

4) In MVC based web apps, we have to use a Servlet as Controller and a set of JSP pages as view part.

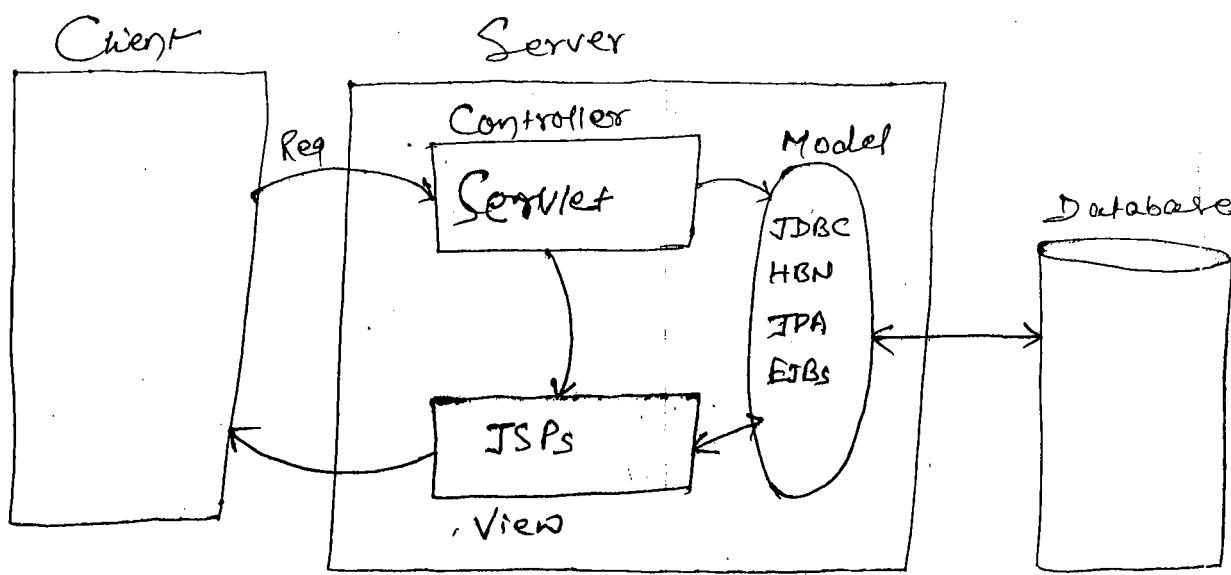
Note (1) Struts is MVC based framework, where ActionServlet is used as Controller and a set of JSP pages are used as view part.

Note (2) :- JSF is MVC based framework, where facesServlet is

⑤ If we perform modifications on Servlets then we have to perform recompilation and reloading onto the server explicitly.

If we perform modifications on existed JSP page then it is not required to perform recompilation and reloading onto the server.

⑥ In case of Servlets, there is no separation between presentation logic and business logic. In case of JSP pages there is a separation between presentation logic and business logic.



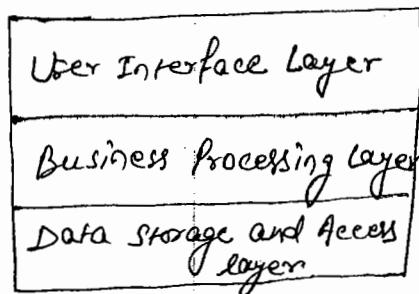
Enterprise:-

It is a business Organisation, a group of Organisation running under single label is called as an Enterprise.

Enterprise Application:-

It is a software application designed for an enterprise in order to simplify their internal business processing.

To prepare an Enterprise application, we have to provide the following layers.

① User Interface Layer:-

This layer will improve look and feel for the enterprise application. ~~This~~

This layer is an entry point for the Customers in order to interact with the enterprise application.

→ This layer will provide very good environment to accept data and to get data from users in order to execute enterprise appn.

→ This layer will provide very good environment to perform Client Side data validations by executing java script functions.

→ To prepare this layer we have to use a separate logic called as presentation logic.

→ To prepare presentation logic in enterprise applications we have to use a set of technologies like AWT, SWING, HTML, JSP, freemarker, velocity, ...  $\Sigma$

## ② Business Processing Layer:-

- This layer is a heart of the enterprise apps.
  - This layer will provide very good environment to define and execute all the business rules and regulations which are required by the clients actually.
- To prepare Business processing layer we have to use a separate logic called as Business logic.

To prepare Business logic in enterprise Apps, we have to use a set of technologies like Servlets, JSPs depending on the Arch, EJBs, ...

## ③ Data Storage and Access layer:-

- The main purpose of Data Storage and Access Layer is to interact with database from Enterprise Applications in order to perform database operations.
- To prepare Data storage and access layer, we have to use separate logic called as "Persistence logic".
- To provide Persistence logic in Enterprise Applications we have to use a set of technologies like JDBC, EJBs - Entity Beans, Hibernate, JPA, ...

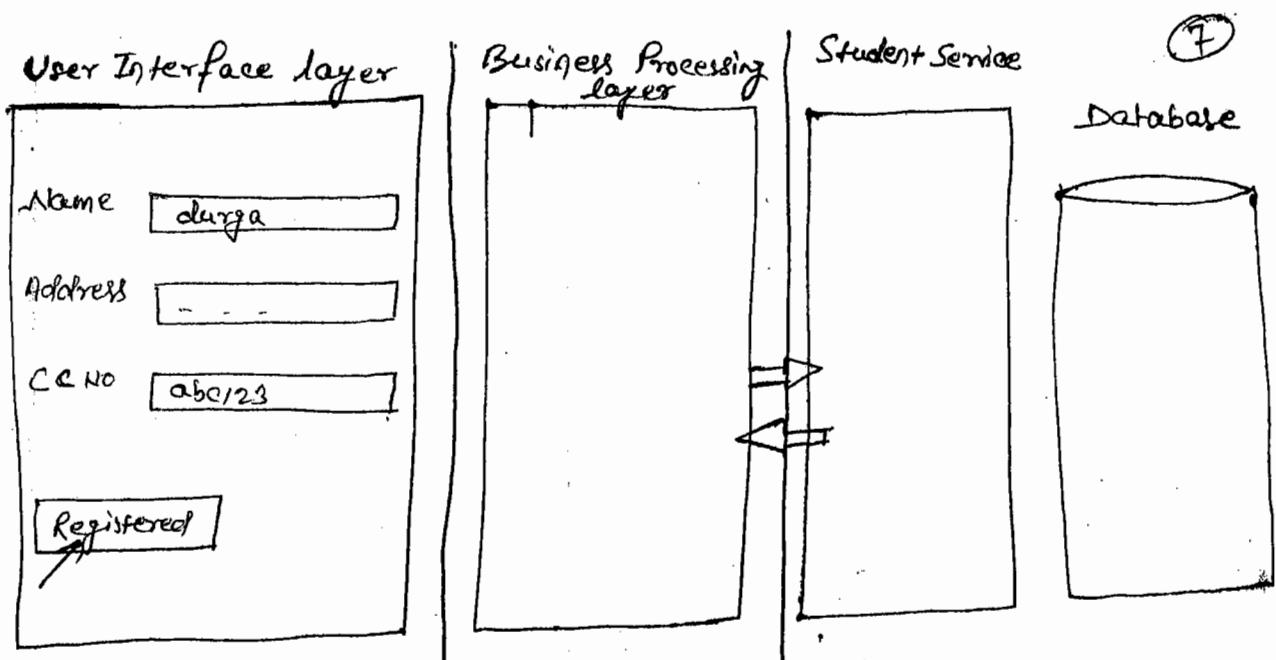


Fig:- Enterprise Application

17-11-2015

### System Architectures:-

- To define logic or level of the Enterprise Applications we have to use System Archs.
- To manage and execute all the logic layers like User interface layer, Business processing layer, Data Storage and Access layer in enterprise Apps we have to use System Archs.

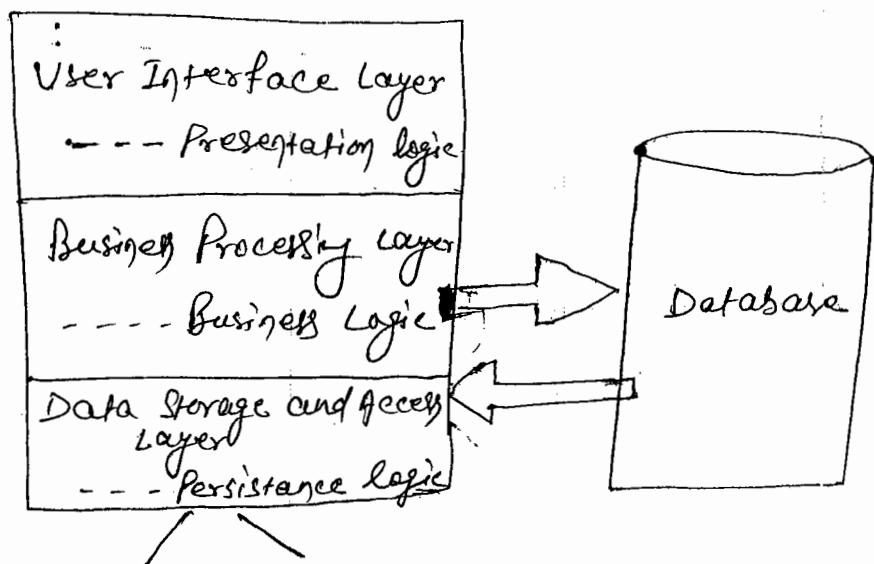
Ex:-

- 1 - Tier Arch
- 2 - Tier Arch
- 3 - Tier Arch
- 
- 
- 7 - Tier Arch

E

## 1-Tier Arch:-

- To prepare enterprise Apps, if we use 1-Tier Arch then we have to execute the complete enterprise App within a single machine i.e. we have to manage User Interface layer, Business Processing layer, Data Storage and Access layer within a single machine



- To prepare enterprise app if we use 1-Tier arch then it's very simple to manage the app, because, the complete application is available within a single machine.
- If we execute the complete application within a single machine then single machine resources may not be sufficient to execute enterprise application, it may reduce application performance.
- In case of 1-Tier arch, there is not clear cut separation between the components which

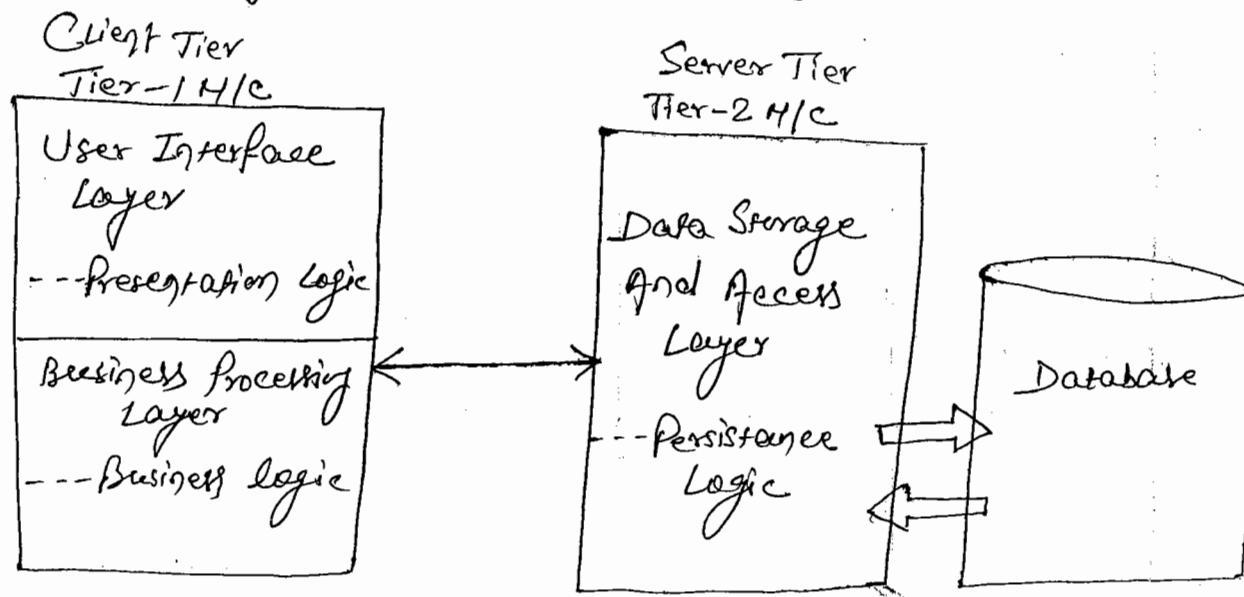
are used for User Interface layer, Business processing layer, Data storage and Access layer, This approach will provide Tightly coupled design, it is not suggestible for enterprise applications.

- 1-Tier will not provide server kind of environment, so that, Components Shareability will not be available.
- 1-Tier Arch will provide single user environment, not multi user environment, it is not suggestible for enterprise apps.
- 1-Tier arch is suggestible for Standalone Applications, not for enterprise Applications.

### 2-Tier Architecture:-

- In 2-Tier Arch, we have to use 2 machines to manage and execute enterprise Appn.
- To prepare enterprise apps, if we use 2-Tier Arch then we have to manage user interface layer and Business processing layer in Tier-1 machine and we have to manage data storage and Access layer in Tier-2 Machine, this approach will provide loosely coupled design in enterprise apps when compared with 1-Tier arch.
- 2-Tier arch will improve database shareability in enterprise apps.
- 2-Tier arch is Conventional (Minimum Arch) Arch to prepare enterprise apps.
- ~~2-Tier arch is not providing multi user environment, so that, Component shareability is not available.~~

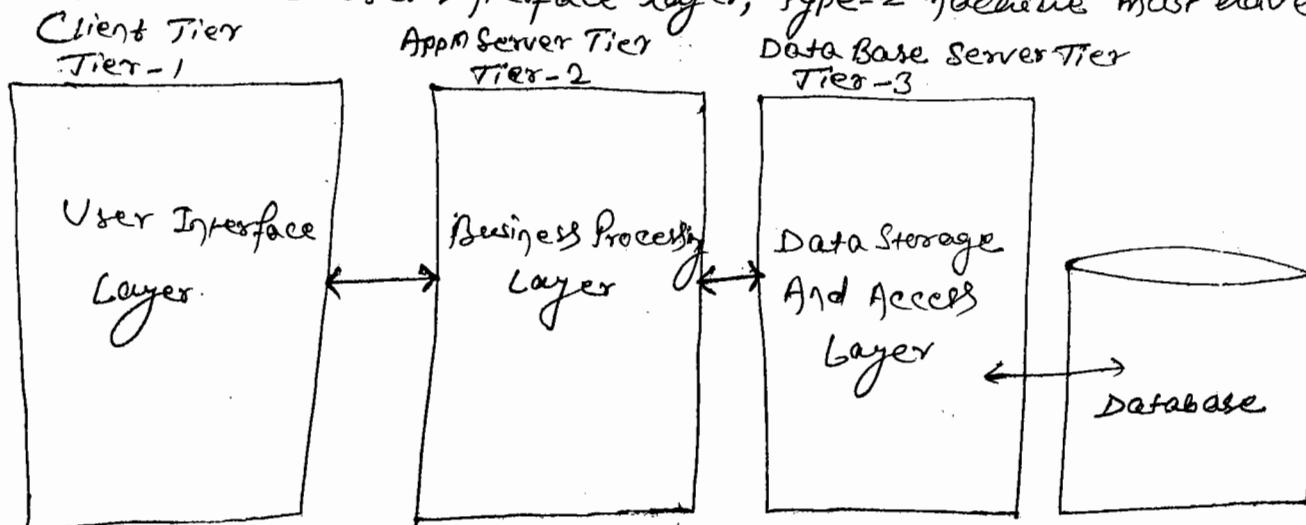
→ 2 Tier arch is for providing separation between User interface layer and Business processing layer.



Note:- Client server architecture is the best example for 2-Tier architecture.

### 3-Tier Architecture:-

→ To design enterprise apps, if we use 3-Tier arch then we have to use 3 types of machines, where type-1 machine must have User interface layer, type-2 machine must have



(9)

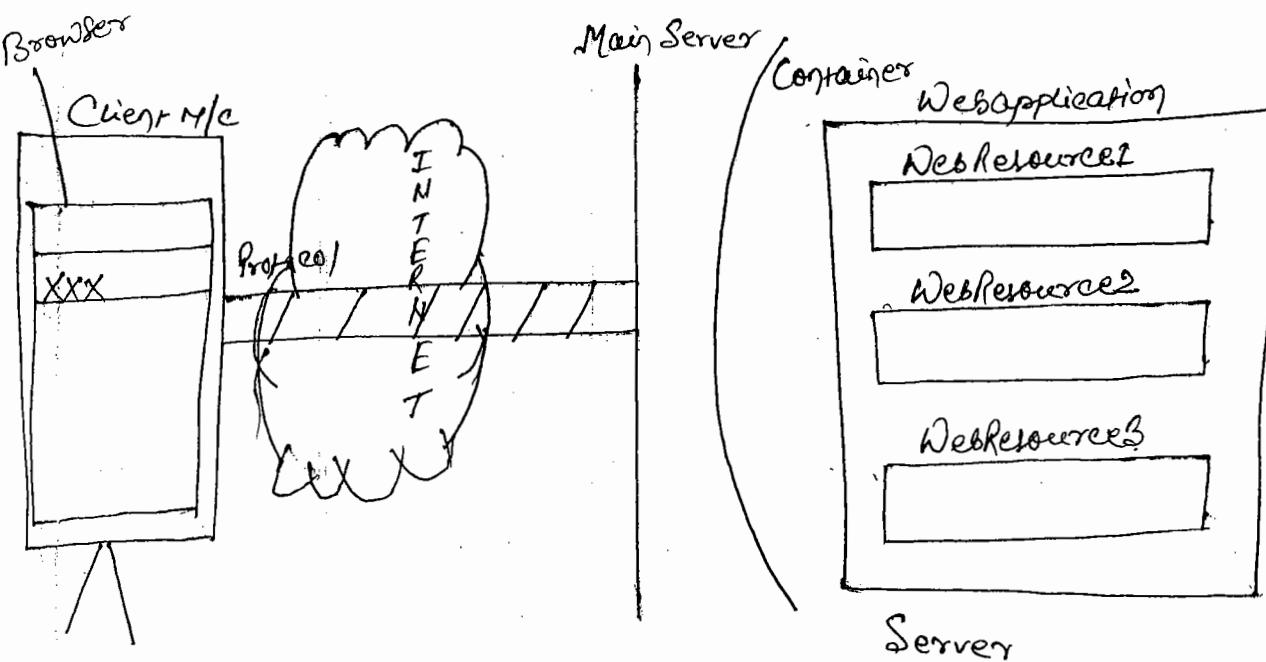
Business Processing layer and Type-3 machines ~~do~~ must have data Storage and Access layer.

- 3-Tier arch. will provide more loosely coupled design when compared with 2-Tier arch. and 1-Tier arch.
- 3-Tier arch. will provide component sharability and database sharability.
- 3-Tier arch. will provide multi-user environment.
- 3-Tier arch. is frequently used arch. in enterprise app development.

19/11/2015

### Client-Server Architecture:-

- Client-Server arch. is <sup>the</sup> conventional arch. to design web apps.



From the above Client-Server Arch, there are three main Components.

- ① Client
- ② Protocol
- ③ Server

### ① Client:-

Client is a web browser, it can be used to send requests to server and it can be used to get responses from server.

→ To send requests from Client to Server we have to provide a String at Client's browser's address bar that is URL [Uniform Resource Locator].

There are two types of URIs:-

- ① URI
- ② URL

Q: What is the diff. b/w URI, URN, & URL?

~~URI~~ is a string Specification provided at client address bar, it can be used to refer a particular resource available at server.

URN is a string ~~Specification~~ provided at client address bar, it can be used to refer a particular server side resource through its logical name.

Note:- In case of services, logical name is anyone specified along with <service-name> tag in web.xml file.

## Servlet

19/11/2015 (13)

→ URL is a String Specification provided at Client address bar, it can be used to refer a particular Server side resource through its locator.

Note:- In case of servlets, locator is a value or pattern provided along with <url-pattern> tag in web.xml file.

Note:- Almost all the servers are supporting URL notation, not URL notation.

### URL Syntax:-

Protocol://Server\_IP-Addr:server\_Port\_No/App\_Path/Resource\_Path  
[?key1=val1&key2=val2....]

Ex:- http://122.121.98.99:8080/regapp/reg?uname=abc&uaddr=Hyd.

→ If server is available in the same Client machine then it's possible to specify "localhost" as server\_IP-Addr.

Ex:- http://localhost:8080/regapp/reg?uname=abc&uaddr=Hyd.

20/11/2015

Q\* What is the difference b/w IP address and port no?

⇒ IP Address is an unique identity to each and every ~~one~~ machine over the network, it would be provided by Network manager at the time of network configuration.

Port no. is an unique identity to each and every process being executed in a machine, it would be provided by local Operating System.

20/11/2015

Q. What is Query String and what is the purpose of Query String in web applications?

⇒ Query String is a set of key-value pairs provided along with URL at Client address bar, it can be used to provide some input data to the Server side resource in order to perform server side action.

Eg:-

http://localhost:8080/loginapp/login?uname=~~abc~~&upwd=abc123.

Protocol:-

Protocol is a set of rules and regulations, it can be used to carry the data from one machine to another machine over the network.

The main role of the Protocol in web applications is to carry request data from client to server and to carry response data from server to client.

Eg:- HTTP, TCP, IP, SMTP, ARP, RARP, ...

Q. What are the requirements to use HTTP protocol in web app?

⇒ In web applications, we need a protocol, it must be ~~connection-oriented~~ - ~~address~~

(1) Connection less:- Not to have physical connection, to have logical connection.

(2) Stateless:- Not to remember clients previous requests data at the time of processing later requests.

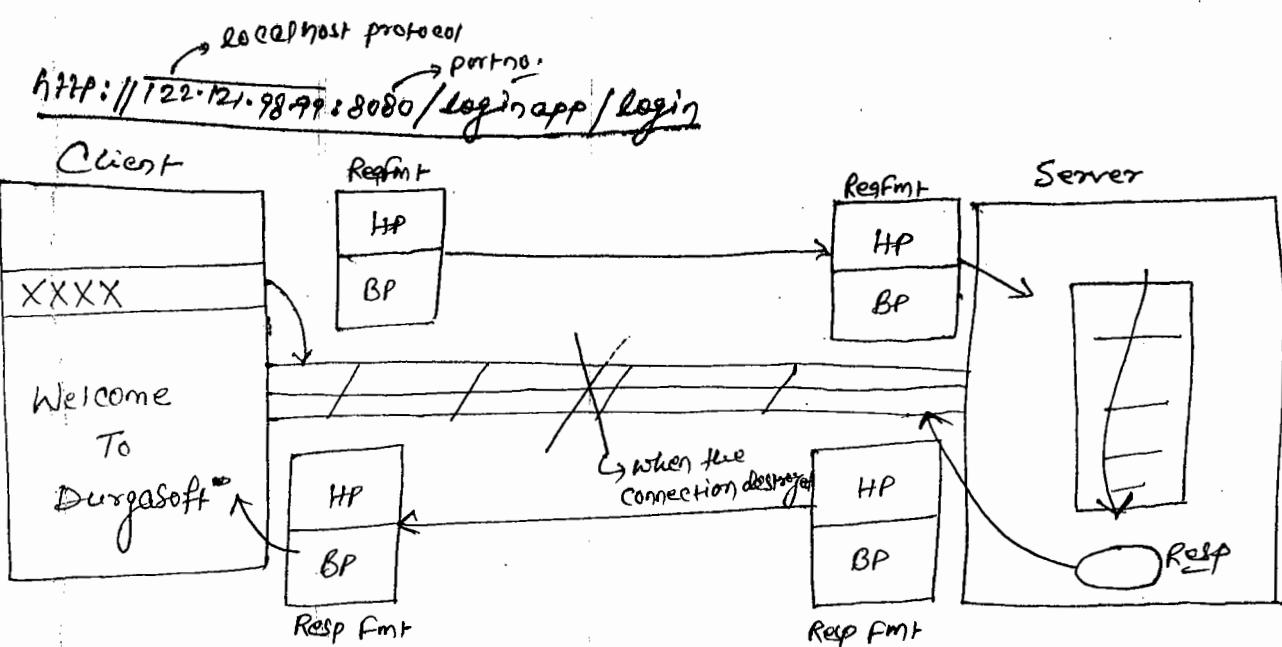
(3) Compatible protocol with Hyper text data:-

In general, in web applications, we will send request data from Client to Server and response data from Server

(15)

to client in the form of hyper text data, so we need a compatible protocol to carry hyper text data between Client and Server.

Among all the protocols, HTTP protocol is having all the above features, so, that we need HTTP protocol in web applications.



21/11/2015

Q How HTTP protocol is able manage its Stateless nature?

⇒ When we submit request from Client to Server, Protocol will take Request and protocol will perform the following actions.

- ① Protocol will establish a Virtual socket Connection between Client and Server on the basis of Server IP address and Server port number.
- ② Protocol will create Request format with header part and body part, where header part is able to manage request headers data that is metadata of Client browser like name of the browser, language and country.

21/11/2015

Setup in browser, zipping formats which are supported by client, encoding mechanisms are supported by client, ... And body part is able to manage request parameters data that is the data which is provided by user at client browser in user forms.

- ③ After creating request format, protocol will transfer request format to server.

Note:- With the above steps, Request is reached to server, where server will identify requested resource, Server will execute requested resource and server will generate response to client.

When response is generated to client, protocol will take that response and protocol will perform the following actions.

- ① Protocol will create response format with header part and body part, where header part is able to manage metadata of the generated dynamic response like size of the response, type of the response, ... and body part is able to manage the actual dynamic response generated by server side resource.
- ② When Response format is created, Protocol will carry response format to client, where Client will receive response.
- ③ When response is received to Client browser, Protocol will destroy the logical connection between Client and Server.

E

23/11/2015

(16)

Protocol is able to remember the present request data upto the time connection is existed. If connection is terminated then protocol will not manage the present request data. Due to this reason HTTP protocol is unable to manage clients previous requests data at the time of processing later requests, this nature of HTTP protocol is called as Stateless nature.

[Note:- As per the application requirements, if we want to manage clients previous requests data at the time of processing later requests then we have to use a set of mechanisms explicitly called as "Session tracking mechanisms".

→ HTTP protocol is providing flexibility like to specify 7 types of request at Client, it is possible because of 7 no. of HTTP methods provided by HTTP protocol.

→ As per HTTP 1.0, HTTP protocol has provided the following HTTP methods.

- ① GET
- ② POST
- ③ HEAD

As per HTTP 1.1, HTTP protocol has provided the following HTTP methods.

- ① OPTIONS
- ② TRACE
- ③ PUT
- ④ DELETE

3

Note: HTTP 1.1 has provided a reserved HTTP methods that is "CONNECT".

[Note:- Servlet 3.2+ is using HTTP2 version.]

Q. What are the diff. b/w GET request and POST request?

⇒ ① GET request is default request.

POST request is not default request.

② GET request is not having body in request format.

POST request is having body in request format.

③ Due to lack of ~~request~~ body in GET request, GET request will carry data through Request Format header part, it allows very less data upto 256 characters, therefore, GET request is able to carry less data.

POST request is able to carry more data due to the availability of ~~body~~ part in Request Format.

④ If we provide data along with GET request then that data is visible on client address bar as query string along with URL, so that, NO security for the data in GET Request.

If we provide data along with POST Request then data is not visible on client address bar as query string, so that POST Request will provide security for data.

(17)

⑤ In general, GET Request is used to perform download operation.

POST Request is used to perform upload operation.

Note:- To perform download operation over a resource then it is required to send data like resource identifier or resource name,... along with GET Request, so, that GET is able to allow less data from Client to Server in order to perform download operation.

⑥ Book marks are supported by GET Request.

Book marks are not supported by POST Request.

Note:- Supporting or not supporting the above 7 types of request is completely depending on the server which we used.

Almost all the servers are supporting GET and POST Request.

[St] Almost all the servers are not supporting PUT and DELETE Request types.]

③ HEAD:-

The main purpose of HEAD Request type to get HEADER part of the requested resource that is metadata of the dynamic response like name of the resource, Content type of the resource, Content size of the resource,...

====

24/11/2015

#### ④ OPTIONS:-

This request type can be used to get the HTTP methods which are supported by the present server available at Server machine.

=

#### ⑤ TRACE:-

The main purpose of this request type is to check whether the server side resource is working fine or not. This request type is performing its action like "Echo Server".

#### ⑥ PUT:-

In web applications, POST and PUT Requests are used to upload data to Server.

To upload a resource to Server if we use POST request then it is not required to specify server side location along with request.

To upload resource to Server, if we use put request then it is mandatory to specify server side location along with request.

=

Note:- PUT request is able to perform resources Overriding at Server but POST request is not performing resources Overriding at Server.

#### ⑦ DELETE:-

This request type can be used to delete a particular resource from Server.

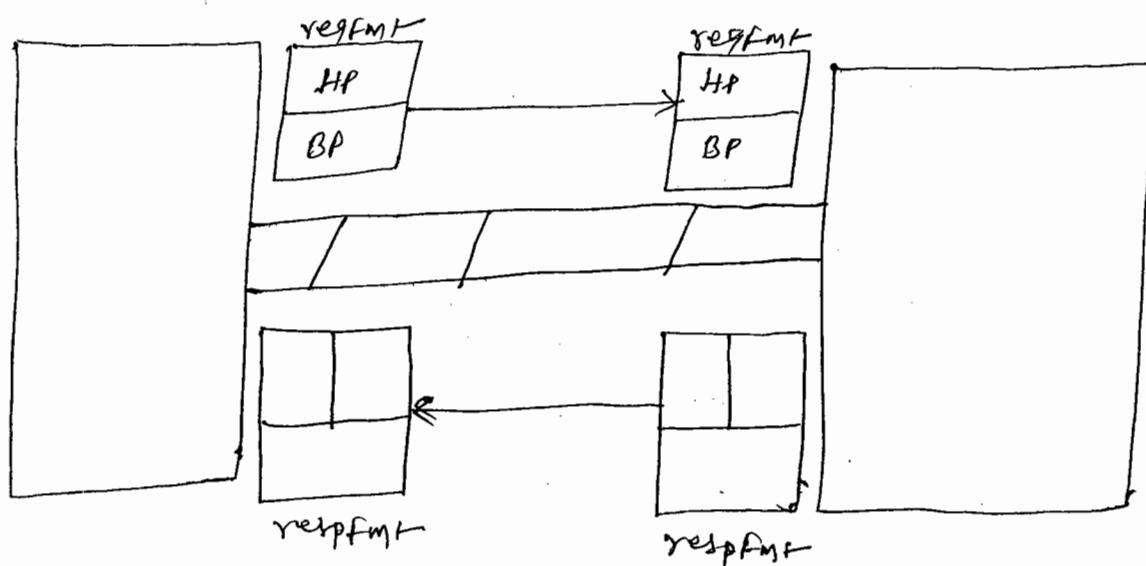
==

## Status Codes:-

Status Code is a no. representation provided by HTTP 1.1 version to represent the status of current request processing. HTTP 1.1 has provided the following Status Codes.

1xx	→ 100 to 199	→ Informational SC
2xx	→ 200 to 299	→ Success Related SC
3xx	→ 300 to 399	→ Redirectional SC
4xx	→ 400 to 499	→ Client Side Errors SC
5xx	→ 500 to 599	→ Server Side Errors SC.

When Server is generating response, Server will provide a particular Status Code value at "Status Line Field" in response format. When Client receive response format, first, Client will take Status Code value from Status Line field available at Response format header part and Client Browser will prepare itself to get Response from Response format.



### ③ Server:-

Server is a special software installed at server machine.

Ex:-

Tomcat

WebLogic

JBoss

Glassfish

WebSphere

JRUN

JEE

- - -

- - -

In Client-Server application, the main role of the server is hold web applications, to get requests from Client, identifying and executing server side resources and generating dynamic response to client.

All the servers are divided into the following two types.

① Web Servers

② Application Servers.

Q. What are the diff. b/w Web Server and Application Server?

⇒ Web Server will provide very good environment to execute web applications.

Application Servers will provide very good environment to execute both web applications and distributed applications.

② Web Servers are not providing middleware services like JNDI, JTA, JCA, JASS, JMS, ... by default.

25/11/2015

(19)

Application servers are providing all the <sup>above</sup> specified facilities.

-ware services by default.

When we install Server software in Server machine then server software is available in the form of the following two methods.

① Main Server

② Container.

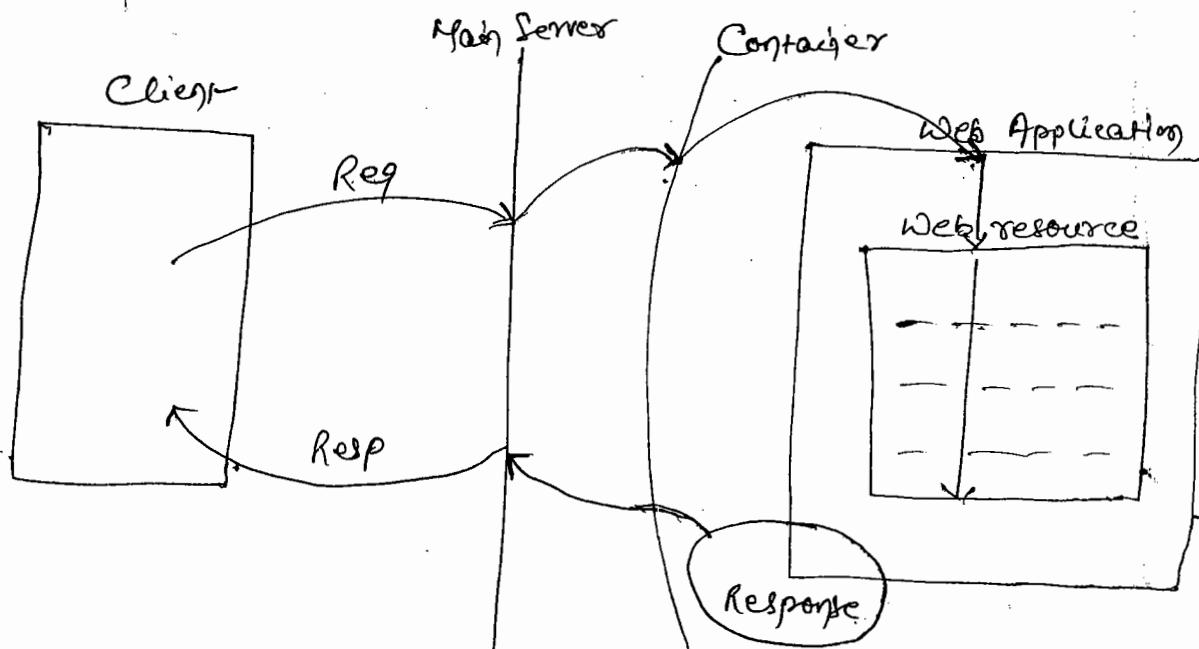
Q. What is the diff. between Main server and Container?

→ When we submit request from Client to Server, at Server machine, Main server will take request, Main server will check whether request is valid or Invalid, if the request is valid then Main server will forward request to Container.

When request is coming to the Container from main server, Container will identify the requested resource, Container will execute the requested resource, Container will generate dynamic response and Container will forward response to main server, where Main server will dispatch response to Client through protocol.

Note:- All the Server side components are executed by following lifecycle actions or lifecycle methods.

Note:- In Server Software, Only, Container is having Server side Components lifecycle implementations, not with Main server, so, that Container is responsible to execute all the server side Components like Servlets, JSPs, ...



### Containers Classification:-

In general, Containers are classified in the following two ways.

① On the basis of the technology which we used to prepare server side components there are two types of containers.

① Web Container:-

It will execute only web components like Servlets, JSPs, ...

② EJB Container:-

It will execute only EJB components.

② On the basis of the containers physical consistency with main server there are three types of containers.

① Standalone Container:-

It is an integration of main server and Container as a single unit.

(2) In-Process Container:-

It is a container existed inside main server.

(3) Out-of-Process Container:-

It is a container existed in out side of the main server.

Steps To Prepare First Web Application:-

26/11/2015

(1) Install Server Software.

(2) Prepare Web Application Directory Structure.

(3) Prepare Deployment Descriptor [Web.xml]

(4) Prepare Web Resources like Servlets, JSPs, ...

(5) Start Server

(6) Deploy web application.

(7) Access web application.

(1) Install Server Software:-

(1) Download apache-tomcat-8.0.9.exe file from internet.

(2) Double Click on apache-tomcat-8.0.9.exe file

(3) Click on Yes button.

(4) Click on Next button.

(5) Click on I Agree button.

(6) Select all the checkboxes (including host manager and examples). Then Click on Next button.

→ Server shutdown port - 8005

→ HTTP/1.1 Connector port - 8080 (Except 8080, we can use any port).  
because of, 8080 port is  
using by Oracle already.)

→ AJP/1.3 Connector port - 8009

→ Windows Service Name - Tomcat8

→ Tomcat Administrator login → user name - durga

(optional)

→ password - durgadeviga.

26/11/2015

→ Roles - admin-gui, manager-gui

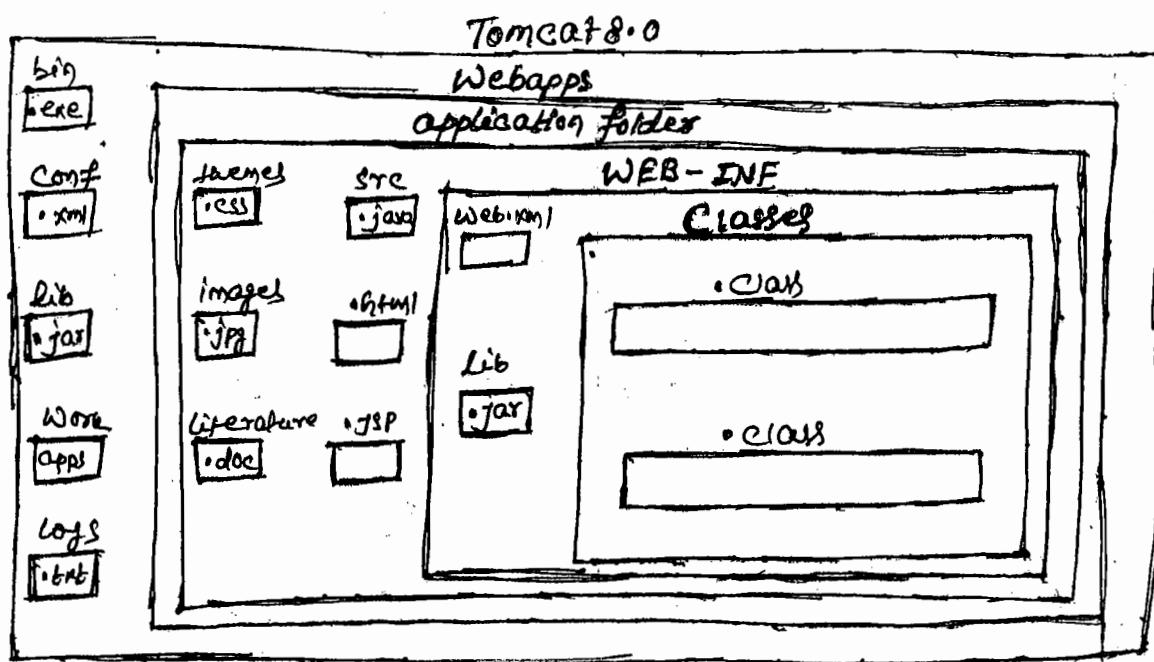
- ⑦ Click on next button.
- ⑧ Select javaSE JRE (if there is java8 by default it will take otherwise we have to select javaSE JRE).
- ⑨ Click on next button.
- ⑩ Change Tomcat installation location from

C:\Program Files (x86)\Apache Software Foundation\Tomcat 8.0  
location to "C:\Tomcat 8.0".

- ⑪ Click on Install button.
- ⑫ De-select Run Apache Tomcat and Show Readme, then click on Finish button.

## ② Prepare Web Application Directory Structure:-

If we want to prepare web application at server then we have to use the following directory structure.



The above web appn structure is divided into the following two parts.

### ① Public Area/Client Area:-

It is an area come under Application folder and outside of WEB-INF folder.

If we keep any resource like HTML or JSP in public area then client is able to access that resource by using its name directly.

HTTP://localhost:1010/loginapp/loginform.html

### ② Private Area/Server Area:-

It is an area come under WEB-INF folder.

If we keep any resource like servlets, filters, listeners under private area then client is unable to access that resources by specifying their names in URL, where to access these resources we have to define alias names or URL-patterns or locators in web.xml file w.r.t. the servlets, filters then we have to access these resources by using aliasnames

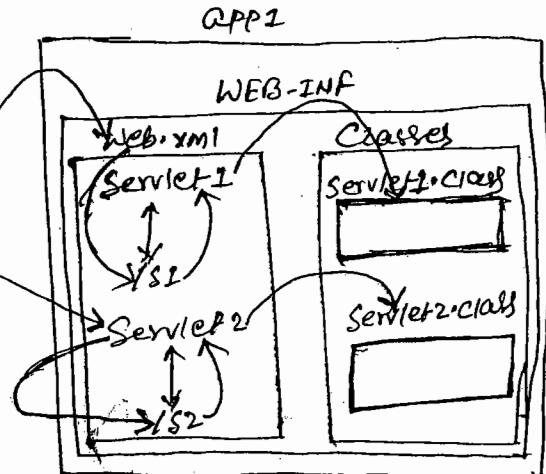
27/11/2015

### ③ Prepare Deployment Descriptor:- [web.xml]:-

http://127:1010/app1/s1

Container

http://127:1010/app1/s2



27/11/2015

Deployment Descriptor is web.xml file, it will provide some description about our present web application required by the container to identify and execute server side resources.

In general, in web applications, we are able to provide the following configuration details in web.xml file.

- ① Welcome files Conf.
  - ② Display Names Conf.
  - ③ Context parameters Conf.
  - ④ Servlets Conf.
  - ⑤ Filters Conf.
  - ⑥ Listeners Conf.
  - ⑦ Initialization Parameters Conf.
  - ⑧ Error page conf.
  - ⑨ Session Time Out Conf.
  - ⑩ Load on Startup Conf.
  - ⑪ Tag Libraries conf.
  - ⑫ Security Constraints conf.
- -----  
-----

total 27-30 No. of conf.  
but these are more valuable,

(22)

In general, in web applications, we will create services under classes folder that is private area, so that, Client is unable to access services by using services name directly. In this context, to execute services, we have to define an url pattern for the service in web.xml file, through the provided url pattern only we have to access services from Client.

In the above context, to define url pattern for services and to provide mapping between url patterns and services we have to use Service Configuration in web.xml file.

To provide Service Configuration in web.xml file we have to use the following XML tags in web.xml file.

### Web.xml

<Web-app>

---

<Servlet>

<Servlet-name> logical name <

<Servlet-class> Fully Qualified name of the servlet

Class </Servlet-class>

</Servlet>

<Servlet-mapping>

<Servlet-name> same logical name <

<url-pattern> /pattern\_name <

<url-pattern> /pattern\_name /url-pattern </url-pattern>

---

</Servlet-mapping>

---

<Web-app> ==

✓

Ex:-

Web.xml

<Web-app>

<Service>

<Servlet-name> ls </Servlet-name>

<Servlet-class> LoginServlet </Servlet-class>

</Service>

<Servlet-mapping>

<Servlet-name> ls </Servlet-name>

<url-pattern> /login ~~<url-pattern>~~ </url-pattern>

</Servlet-mapping>

</Web-app>

====

Note:- Upto Servlets 2.4 version, we are able to provide only one url pattern for one service but from Servlet 2.5 version onwards we are able to declare more than one url pattern for a single service.

Q Is it mandatory to provide web.xml file in web applications?

⇒ Case 1:-

If you are not using the server side components like services, filters, listeners, ... in web applications then web.xml file is mandatory or optional it's totally depending on the type of server which we used.

If we use the servers like Tomcat then web.xml file is optional.

If we use the servers like weblogic then web.xml file is mandatory. (23)

### Cases:-

If we use the server side components like servlets, filters, listeners, ... in web application then web.xml file is optional or mandatory is completely depending on the servlets versions which we are using.

If we use servlet 2.5 and below then web.xml file is mandatory.

If we use servlets 3.0 or above versions then web.xml file is optional, instead of web.xml file we are able to use "Annotations".

28/11/2015

Q In web applications, is it possible to change the name and location of deployment descriptor?

→ No, it is not possible to change the name and location of deployment descriptor, because container was implemented in such a way that to search deployment descriptor with the fixed name that is web.xml and with the fixed location that is WEB-INF folder.

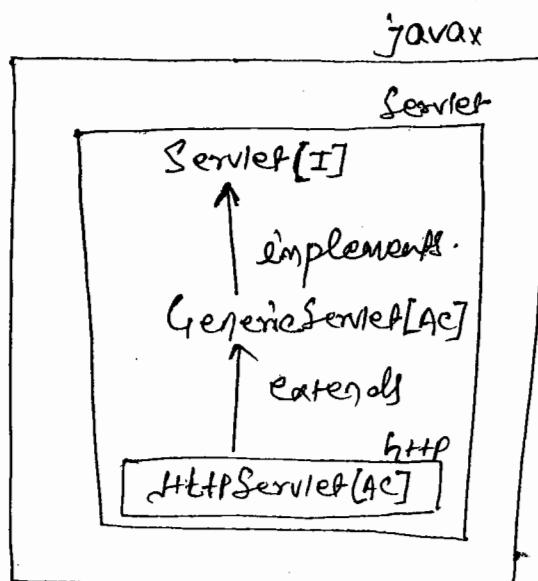
(Note:- In the web application frameworks like Struts, JSF, ... deployment descriptor name and location is fixed but the name and location of configuration files are varied.)

Z

28/11/2015

- Q) Prepare the web resources like Servlets, JSPs, ... as per the requirement :-

To prepare Servlets, Servlet API has provided the following predefined classes and interfaces in javax.servlet package and in javax.servlet.http package.



- Q) What is Servlet and in how many ways we are able to prepare Servlets?

→ Servlet is a resource or an object available at server which is managed by the container, which is executed to generate dynamic response from server.

Servlet is an object available at server, which must implement javax.servlet.Servlet interface either directly or indirectly.

As per the predefined library provided by Servlet API there are three approaches to prepare Servlets.

## ① Implementing servlet interface:-

In this approach, we have to define (declare) an user defined class, it must implement `[javax.servlet.Servlet]` interface.

```
public class MyServlet implements Servlet
```

— implementation to all the servlet interface methods —

## ② Extending GenericServlet abstract class:-

In this approach we have to declare an user defined class, it must be extended from

`[javax.servlet.GenericServlet abstract class]`

```
public class MyServlet extends GenericServlet
```

— Overriding service provider methods —

## ③ Extending HttpServlet abstract class:-

In this approach, declare an user defined class, it must be extended from `[javax.servlet.http.HttpServlet abstract class]`.

```
public class MyServlet extends HttpServlet
```

— Override Service provider methods —

====

## ⑤ Start Server:-

There are two ways to Start Server.

① Double Click of "Tomcat" application available at "C:\Tomcat 8.0\bin" location.

② Use Tomcat system service by using the following path.

in Windows OS:-

Start → Run → Write "services.msc" in search bar → Select

"Apache Tomcat 8.0 Tomcat" service → Select the required  
hyperlink or icon like "Start Service, Stop Service, pause  
Service, restart Service" as per the requirement.

## ⑥ Deploy Web application:-

If you prepare web applications under "webapps" folder  
then container will deploy all the web applications  
automatically to server space.

## ⑦ Access Web application:-

After deploying web applications, if we want to access  
web application then we have to provide the following  
URL at client address bar.

`http://localhost:1010/app1/first`

where app1 is application name and "first" is url pattern  
for FirstServlet.

## Servlets Design:-

There are three approaches to prepare Servlets.

- ① Implementing `Servlet` interface
- ② Extending `GenericServlet` abstract class
- ③ Extending

### ① Implementing `Servlet` interface:-

In this approach, we have to declare an user defined class, it must be implement `javax.servlet.Servlet` interface.

```
public interface Servlet {
    public void init(ServletConfig config) throws ServletException;
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException;
    public ServletConfig getServletConfig();
    public String getServletInfo();
    public void destroy();
}

public class MyServlet implements Servlet
```

— implementation for all the methods of `Servlet` interface —

### init(...):-

This method can be used to perform `Servlet` initialization.

To access `init()`, Container must create `ServletConfig` object with all the `Servlet` configuration details which we specified in web.xml file.

## Service (---, ---)

- This method is like math() method, in J2EE applicat-  
- ONS, it will include business logic which we want to  
execute upon receiving request from client and it will  
be executed by the container automatically when it  
receive request from client.

To access this method, container must create ServletReq-  
uest object and ServletResponse object, where ServletReq-  
uest object is able to store request data and ServletResponse  
object is able to store dynamic response provided by  
Service() method.

## getServletConfig()

- This method will return ServiceConfig object which is created  
by the container as part of Servlet Initialization.

## getServletInfo()

- This method will return information about Servlet.

## destroy()

- This method will perform Servlet Deinstantiation.

Note:- from the above methods, init(), service() and  
destroy() methods are called as Lifecycle methods,  
because, these methods are used directly in Servlet  
Lifecycle. The methods getServletInfo(), getServletConfig()  
are called as Non-Lifecycle methods, which are not  
used in Servlet Lifecycle.

## Steps to implement Service() method:-

### (1) Set Content Type In Response Header part:-

To Set Content type in Response header part we have to use the following method from `ServletResponse`.

```
public void setContentType(String resp-type)
```

Where `resp-type` may be `text/html`, `img/jpg`, ...  
Ex:- `response.setContentType("text/html");`

The main intention to specify Content type in response format header part is to give an information to the client about generating the type of response that present servlet is generating.

Note:- The default Content type in web applications is "text/html".

### (2) Get PrintWriter object:-

The main intention to get `PrintWriter` object is to send response to response object in order to submit response to Client.

To get `PrintWriter` Object we have to use the following method from `javax.servlet.ServletResponse`.

```
public PrintWriter getWriter()
```

Ex:- `PrintWriter out = response.getWriter();`

### ③ Send response data to PrintWriter :-

To send response data to PrintWriter we have to use the following method.

public void printIn(String data)

Eg:- Out.printIn("<html>");

Out.printIn("<body>");

Out.printIn("Hello Client");

Out.printIn("</body></html>");



firstapp

|- WEB-INF

  |- Web.xml

|- Classes

  |- WelcomeServlet.java

  |- WelcomeServlet.class

#### Web.xml

<web-app>

  < servlet >

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

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

  < /servlet >

  < servlet-mapping >

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

```

<url-pattern>/welcome</url-pattern>
</servlet-mapping>
</web-app>.

```

### WelcomeServlet.java

```

import javax.servlet.*;
import java.io.*;
public class WelcomeServlet implements Servlet {
    public void init(ServletConfig config) throws ServletException {
    }
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<font color='red' size='7'>");
        out.println("<b>");
        out.println("Welcome To Durga Software Solutions");
        out.println("</b><br><font></font></body></html>");
    }
    public ServletConfig getServletConfig() {
        return null;
    }
}

```

```
public String getServletInfo()
{
    return "";
}

public void destroy()
{
}
```

→ C:\Tomcat8.0\webapps\firstapp\WEB-INF\classes> Set Classpath =  
→ javac WelcomeServlet.java C:\Tomcat8.0\lib\Servlet-api.jar

C:\Tomcat8.0\Websapps\firstapp\WEB-INF\classes>ls -l

## Servlet Flow of Execution:-

When we start server the role of the container is,

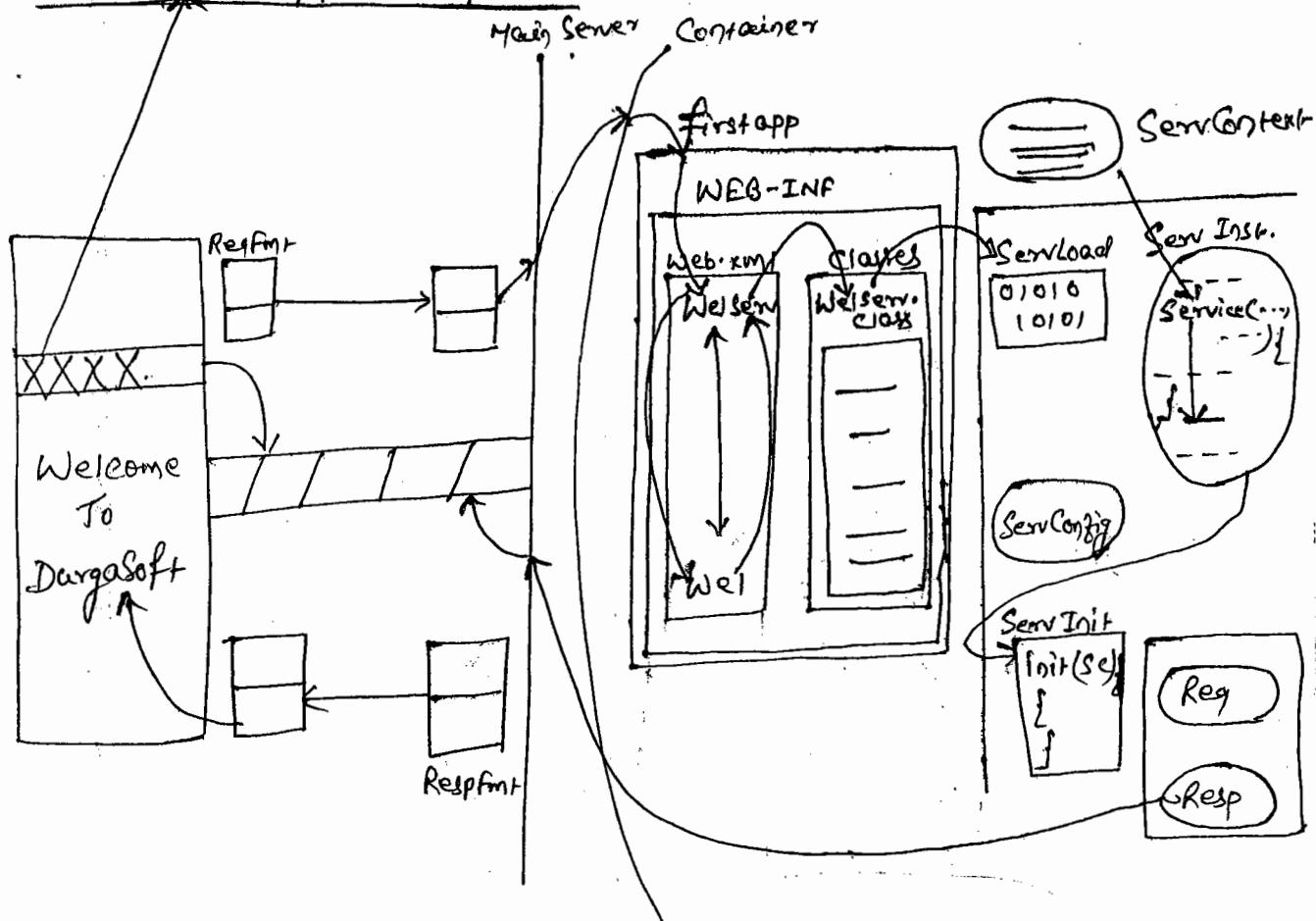
- ① Recognize all the web apps available under webapps folder.
  - ② Deploying all the web apps into server space.
  - ③ Creating a separate object for each and every deployed application called as "ServletContext" object.

When Container recognize web applications, container will identify web.xml file under WEB-INF folder at each and every app. Then container will perform web.xml file loading, parsing and reading the content of

(28)

Web.xml file. In web.xml file if any application level data is available then Container will store that data in ServletConfig object.

HTTP://127.0.0.1:1010/firstapp/wel.



**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

When we submit request from Client to Server, protocol will take the request and protocol will perform the following actions.

- ① Protocol will establish Virtual Socket Connection between Client and Server as per Server IP address and Server port 90, which we have specified in URL.
- ② Protocol will prepare Request Format with header part and body part, header part is able to store request headers data and body part is able to store request parameters data.
- ③ Protocol will carry request format to Server.

→ Main Server will take request from protocol, Main Server will check whether request is proper or not, if it is proper then Main Server will forward request to Container.  
→ Container will take request from Main Server, Container will check whether the resource name is an uri pattern or not, if it is an uri pattern then Container will identify the requested service name and location from web.xml file on the basis of uri pattern, then Container will identify the requested service class under classes folder and perform the following actions to execute Servlet.

### ① Servlet Loading :-

Container will load service class bytecode to the memory.

```
Class C=Class.forName("Welcomeservlet");
```

## ② Servlet Instantiation

Container will Create object for the loaded Servlet class.

```
WelcomeServlet ws=(WelcomeServlet) c.newInstance();
```

## ③ Servlet Initialization :-

Container will create `ServletConfig` object with `Servlet Configuration` details and Container will access `init()` method.

```
ws.init(config);
```

## ④ Creating Request and Response Object :-

After the servlet initialization, Container will Create thread to access `service(..., ...)` method, but to access `service()` method Container has to create `ServletRequest` object and `ServletResponse` object.

## ⑤ Generating Dynamic Response :-

By passing request and response objects, Container will execute `service()` method, with this container will generate dynamic response in Response object.

```
ws.service(request, response);
```

## ⑥ Dispatching dynamic response to Client :-

When Container generated thread reached to ending point of `service()` method then that thread will come to dead state, with this, Container will dispatch dynamic response to Main Server, where Main Server will dispatch dynamic response to protocol, where protocol will perform the following actions.

① Protocol will create Response format with header part and body part where header part is able to store metadata of dynamic response and body part is able to store actual dynamic response.

② Protocol will carry response format to Client.

③ Protocol will terminate virtual socket connection between Client and Server.

⑦ Destroy Request and Response object:—

When Connection is terminated between Client and Server, Container will destroy request and response objects.

⑧ Servlet Deinstantiation:—

After destroying request and response objects Container will go to waiting state depending on the Server which we used, if no requests are coming to the same resource then Container will destroy Servlet object.

`ws.destroy();`

Ex:— Just before destroying Servlet Object, Container will destroy `ServletConfig` object.

⑨ Servlet Unloading:—

Container will remove Servlet bytecode from Operational memory.

⑩ Destructing ServletContext Object:—

When we shutdown the server or when we undeploy the web application Container will destroy `ServletContext` object.

02/12/2015

(30)

### Drawbacks:-

In this approach, if we want to prepare servlets then we have to implement javax.servlet.Servlet interface, where we have to provide implementation for all the methods of javax.servlet.Servlet interface irrespective of the actual requirement. This approach will increase unnecessary methods in service classes.

### ② Extending GenericServlet:-

In this approach, we have to declare an user defined class, it must be extended from javax.servlet.GenericServlet Abstract Class, where we have to override service() method with our implementation.

```
public abstract class GenericServlet implements Servlet, ServletConfig, Serializable
{
    private transient ServletConfig config;
    public void init(ServletConfig config) throws ServletException
    {
        this.config = config;
        init();
    }
    public void init()
    {
        public abstract void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException;
    }
}
```

Cont..

```
public ServiceConfig getServiceConfig()
{
    return config;
}

public String getServiceInfo()
{
    return "";
}

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

~~class~~

```
public class MyServlet extends GenericServlet
{
    public void service(ServletRequest request, ServletResponse response)
        throws ServletException, IOException
```

———— Implementation ——

→ GenericServlet is an abstract class provided by Servlet API in the form of javax.servlet.GenericServlet, it is a direct implementation class to servlet interface and it has provided default implementation for all the methods of servlet interface except service() method.

Note - GenericServlet abstract class has implemented ~~Servlet~~ <sup>(3)</sup> ServletConfig and Serializable ~~object~~ interfaces along with Servlet interface.

In GenericServlet abstract class, init() method is overloaded method.

GenericServlet is an idea coming from "Adaptor Design Pattern" to avoid unnecessary methods implementations while implementing servlets interface in servlet classes.

### GenericServletApp

08/12/2015

genericServletApp

|—WEB-INF

|—Web.xml

|—classes

|—WelcomeServlet.java

|—com.durgasoft.WelcomeServlet.class.



Web.xml

<Web-app>

<Servlet>

<Servlet-name> WS </Servlet-name>

<Servlet-class> com.durgasoft.WelcomeServlet </Servlet-class>

</Servlet>

→ doubt

<Servlet-mapping>

<Servlet-name> WS </Servlet-name>

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

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

</Servlet-mapping>

</Web-app>

3

## WelcomeServlet.java

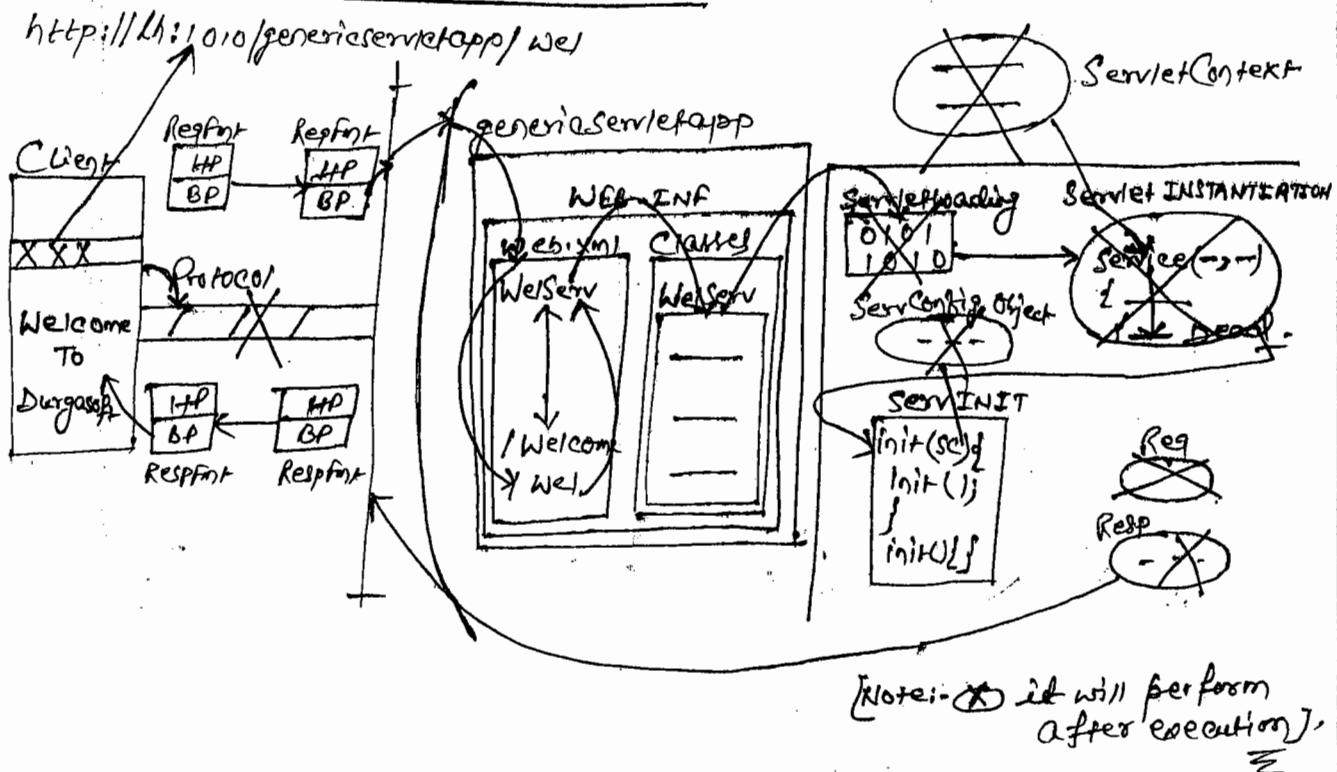
03/12/2015

```
package com.durgasoft;
import javax.servlet.*;
import java.io.*;
public class WelcomeServlet extends GenericServlet
{
    public void service(ServletRequest request, ServletResponse
        response) throws ServletException, IOException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<font color='red' size='7'>"); 
        out.println("<b>"); 
        out.println("Welcome to Durga Software Solutions");
        out.println("</b><br><font></font></body></html>");
    }
}
```

C:\Tomcat 8.0\Websapps\genericservletapp\WEB-INF\classes> set  
Classpath=C:\Tomcat 8.0\lib\Servlet-API.jar;  
classes> javac -d .\ WelcomeServlet.java

3

## Flow of Execution for GenericServlet:-



Note:- In case of first approach [Implementing Servlet interface], Container will execute init(ServletConfig) method in Servlet Initialization phase. In case of second approach [Extending GenericServlet], Container will execute the following two methods as part of Servlet initialization.

~~init(ServletConfig)~~  
~~init()~~

09/12/2015

### ③ Extending HttpServlet Abstract Class:-

In this approach, we have to declare an user defined class, it must be extended from javax.servlet.http.HttpServlet abstract class, where we have to provide doxxx() method of the basis of xxx request type.

```
public abstract class HttpServlet extends GenericServlet  
{  
    public void service(ServletRequest req, ServletResponse resp)  
        throws ServletException, IOException.  
    {  
        HttpServletRequest hreq = (HttpServletRequest)req;  
        HttpServletResponse hresp = (HttpServletResponse)resp;  
        service(hreq, hresp);  
    }  
    protected void service(HttpServletRequest hreq, HttpServletResponse  
        hresp) throws ServletException, IOException.  
    {  
        String method = hreq.getMethod();  
        if (method.equals("GET"))  
        {  
            doGet(hreq, hresp);  
        }  
        if (method.equals("POST"))  
        {  
            doPost(hreq, hresp);  
        }  
    }  
}
```

```

--- -
- - -
if (method.equals("DELETE"))
{
    doDelete(hreq, hresp);
}

```

```

protected void doGet(HttpServletRequest hreq, HttpServletResponse
                      hresp) throws ServletException, IOException
{
}

```

```

protected void doDelete(HttpServletRequest hreq, HttpServletResponse
                        hresp) throws ServletException, IOException
{
}

```

```

public class MyServlet extends HttpServlet
{

```

```

    public void doXXX(HttpServletRequest hreq, HttpServletResponse
                        hresp) throws ServletException, IOException

```

————— implementation —————

3

→ HttpServlet is an abstract class, a child abstract class to GenericServlet abstract class, it contains mainly the following three methods.

### ① Service(ServletRequest, ServletResponse)

- It will take ServletRequest and ServletResponse objects as parameter values.
- It will convert ServletRequest to HttpServletRequest and ServletResponse to HttpServletResponse objects by using downcasting.
- It will access another service() method by passing HttpServletRequest and HttpServletResponse objects as parameter values.

Ex:- Service(freq, fresp);

### ② service(HttpServletRequest, HttpServletResponse)

- It will take HttpServletRequest and HttpServletResponse objects as parameter values.
- It will identify the request type.

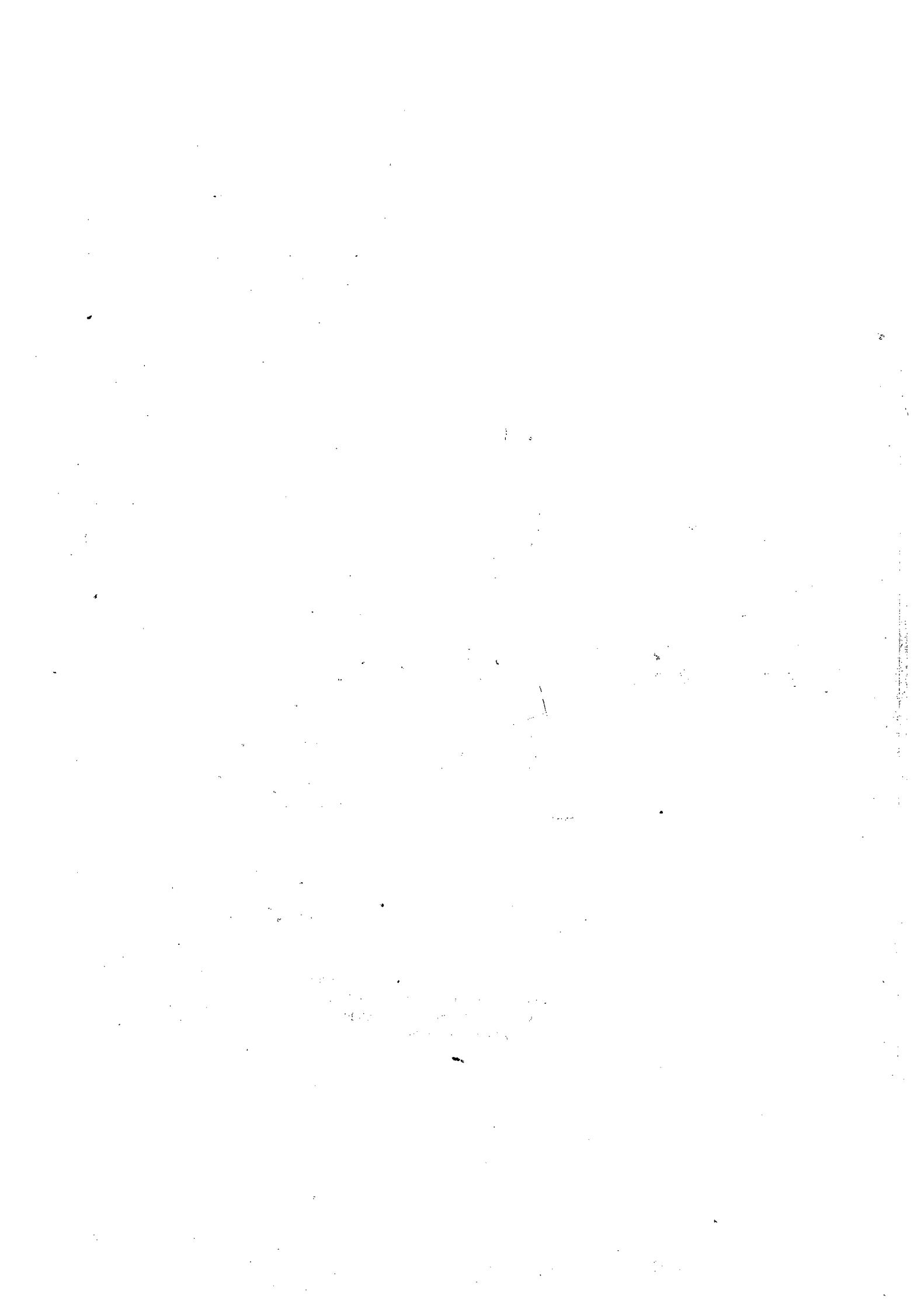
String method = freq.getMethod();
- It will access doXXX(-,-) method on the basis of request type that is "method" variable value.

3

### ③ doXXX(HttpServletRequest, HttpServletResponse)

- It will take HttpServletRequest and HttpServletResponse object as parameter values.
- It must be Overriden by a sub class method in order to create application logic.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.





10/12/2015

Q. Is it possible to override service() method in HttpServlet?

→ Yes, it's possible to override service() method in HttpServlet, but always it is suggestible to override doXXX() methods in HttpServlet depending on the request type.

If we override service() method in HttpServlet then Container will execute user defined service() method, Container will not execute predefined service() method, so that, Container will not be reached to doXXX() method.

Note:- In web apps, if we want to override service() method in servlets then it is suggestible to go for GenericServlet - Let only, no need to come for HttpServlet.

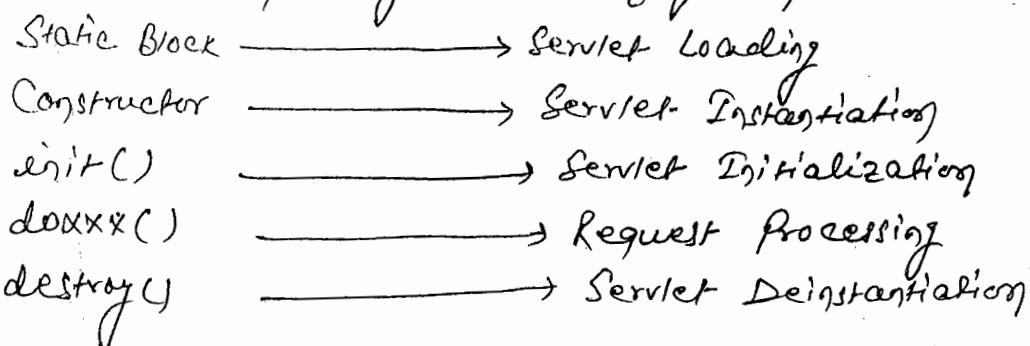
Ex:-

```
public class MyService extends HttpServlet {  
    public void service(ServletRequest req, ServletResponse resp)  
        throws ServletException, IOException {  
        out.println("From service() method");  
    }  
    public void doGet(HttpServletRequest req, HttpServletResponse  
        resp) throws ServletException, IOException {  
        out.println("From doGet() method");  
    }  
}
```

If we submit request to the above servlet then Container will execute service() method only, Container will provide "from service() method" response on Client browser.

Q. Is it possible to provide both constructor and init() method within a single servlet?

→ Yes, it is possible to provide constructor and init() method within a single servlet. If we provide static block, constructor, init() method, doXXX() method, destroy() method in a servlet then container will execute all these elements at the corresponding server lifecycle phase like below.



Ex:-

```

public class MyServlet extends HttpServlet {
    static {
        System.out.println("Servlet Loading");
    }
    public MyServlet() {
        System.out.println("Servlet Instantiation");
    }
    public void init(Config config) throws ServletException {
        System.out.println("Servlet Initialization");
    }
    public void doGet(HttpServletRequest request, HttpServletResponse
                      response) throws ServletException, IOException {
        System.out.println("Request Processing");
    }
    public void destroy() {
        System.out.println("Servlet Deinstantiation");
    }
}

```

If we submit request to the above servlet then container will provide the following response or server prompt.

- Servlet Loading
- Servlet Initialization
- Servlet Initialization
- Request Processing.

Servlet Deinstantiation [When we shutdown server or when we undeploy web application].

If we want to provide constructors inside servlet class then we have to provide only 0-arg constructor and public constructor.

If we provide parameterized constructor in servlet class then container will provide the following exception messages.

javax.servlet.ServletException : Error Instantiating servlet class WelcomeServlet

→ With the root cause:

java.lang.InstantiationException : WelcomeServlet

→ If we provide non-public constructor in servlet class then container will rise an exception like

javax.servlet.ServletException : Error Instantiating servlet class WelcomeServlet

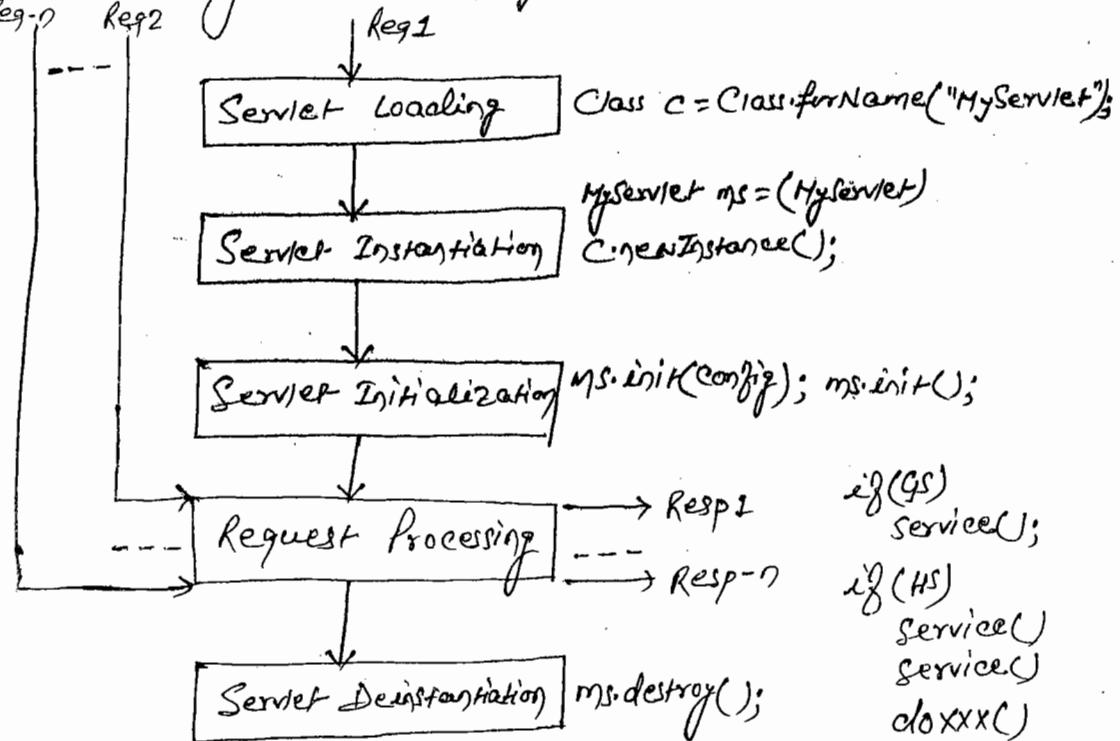
→ With the root cause:

java.lang.IllegalAccessException



## Servlet Lifecycle :-

When we submit request from Client to Server for a particular Servlet then Container will execute that Servlet by using the following Servlet lifecycle actions.

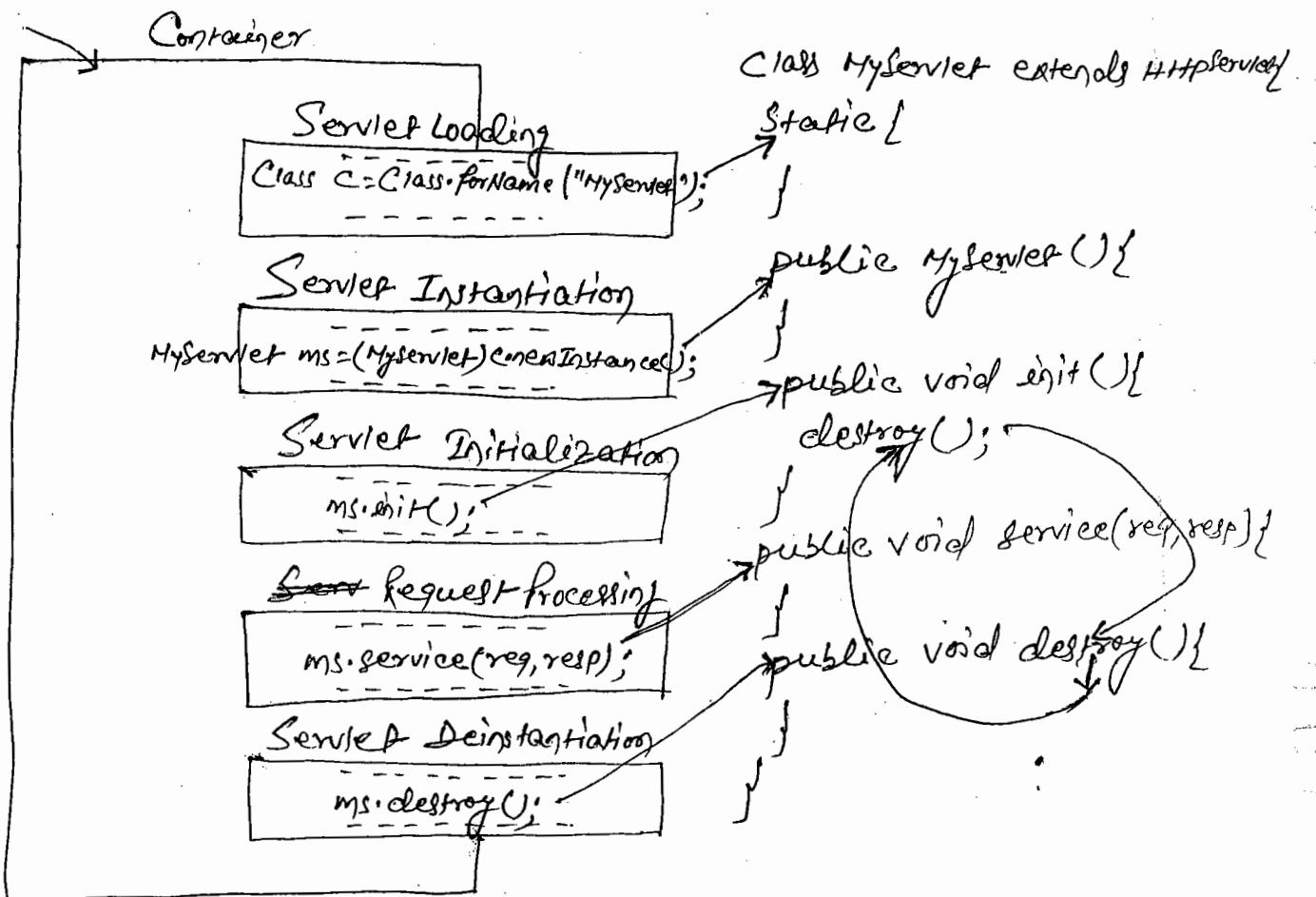


From the above representation, it is possible to send more than one request to a single servlet at a time. If we send more than one request to a single servlet then container will perform Servlet Loading, Servlet Instantiation, Servlet Initialization for only first request, from second request onwards container will bring all the request to Request processing phase.

Q. In Servlet Applications, If we access destroy() method from init() method then is it possible for the Container to perform Servlet Deinstantiation as part of servlet Initialization as per destroy() method call?

→ No, It is not possible for container to perform servlet Deinstantiation for our explicit destroy() method call from init() method.

11/12/2015



Container is a box, it contains implementations for server side components lifecycle actions. Server container is having Servlet Lifecycle actions implementation part.

When request is coming from client to the server container then container will identify servlet class under classes folder and container will execute servlet class by executing servlet lifecycle implementation part internally.

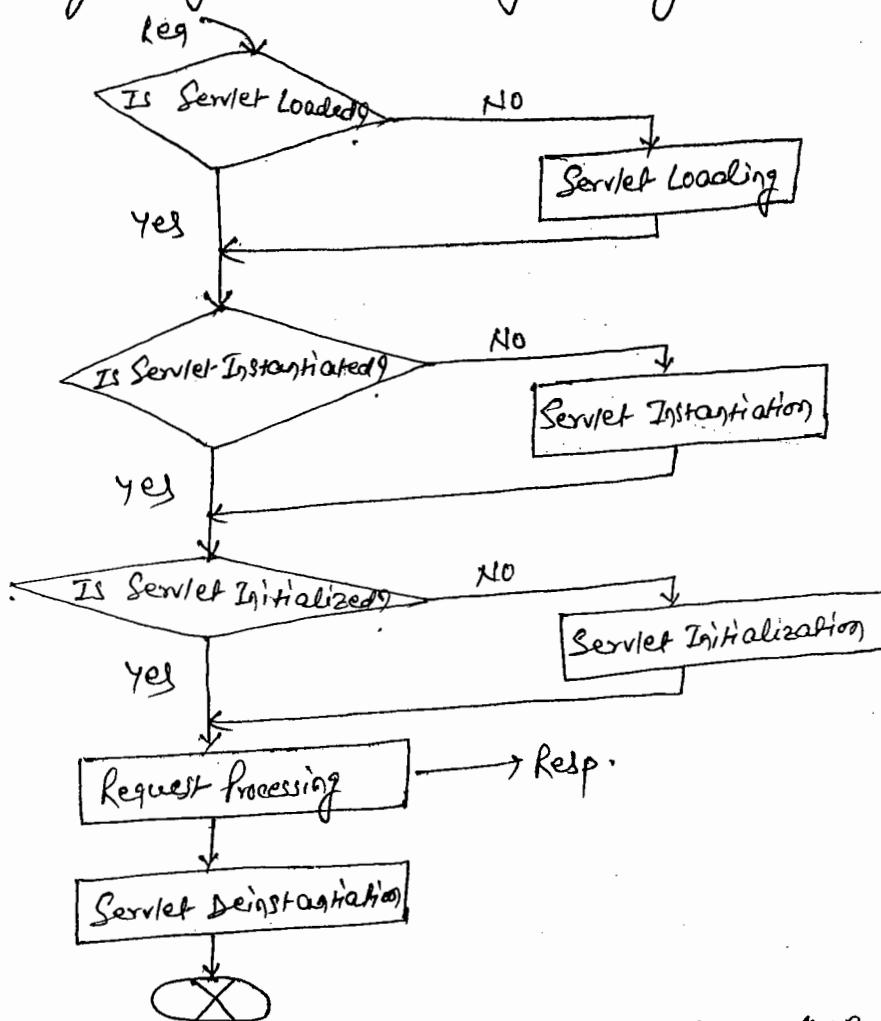
Container will access destroy() method inside servlet Deinstantiation life cycle action to execute the statements which we have provided inside destroy() method.

\* imp statement for this question.

(38)

If we access destroy() method from init() method then Container will execute destroy() method like a normal java method, Container will not execute ~~destroy() method~~ like as lifecycle method

In web applications, if we submit more than one request to a single servlet then Container will process all the requests by using the following lifecycle actions.



In web applications, we are able to submit more than one request to a single servlet, where single servlet is able to process more than one request at a time, that is single servlet is able to handle more than one thread at a time without having data inconsistency, therefore, by default, all the servlet and JSPs are threadsafe.

12/12/2015

As per the requirement, if we want to allow only one request at a time to the servlet then we have to implement javax.servlet.SingleThreadModel marker interface to the servlet class.

public class MyServlet extends HttpServlet implements SingleThreadModel

{

    — implementation —

}

javax.servlet.SingleThreadModel interface is deprecated interface. It is an outdated interface, which is not supported by all the latest versions of the servers.

To achieve the above requirement, we have to use synchronization as an alternative for SingleThreadModel interface.

public class MyServlet extends HttpServlet

{

    public void doxxx(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException

{

        Synchronized(this){

            — implementation —

}  
}

=

## Load-On-Startup Configuration:-

- In general, in web applications, Container will perform a particular Servlet loading, instantiation and initialization after the Server startup and after getting first request from Client.
- As per the requirement, if we want to perform a particular Servlet loading, instantiation and initialization at the time of server startup then we have to provide load-on-startup configuration for the respective Servlet in web.xml file under servlet configuration.
- To provide load-on-startup configuration in web.xml file we have to use the following XML tag.

<web-app>

-----

<servlet>

-----

<load-on-startup> val</

-----

</servlet>

-----

</web-app>

Σ

- Where val may be either 0 or +ve value, it will not allow -ve value.

- In web applications, if more than one servlet is having load on startup configuration in web.xml file then container will load ~~first~~ a servlet which is having less load on startup value and container will load a servlet next which is having high load on startup value.

→ If more than one service is having same load on startup value then container will load all the services in its own ordering, it will not follow service configuration order provided in web.xml file.

Note:- In MVC based web applications, we have to use a service as controller, where ControllerServlet must have no. of responsibilities in its initialization part. If we perform ControllerServlet initialization after the server startup and after getting first request then response time will be increased, it will reduce application performance.

In the above context, if we want to improve performance of server side applications then we have to perform ControllerServlet loading, instantiation and initialization at the time of server startup, for this, we have to provide <load-on-startup> configuration under service configuration in web.xml file.



### ServletConfig:-

ServletConfig is an object, it able to store all the configuration details of a particular service which we specified in web.xml file, where service configuration details are logical name of the service, initialization parameters, ....

- ServletConfig is an object, it will provide the complete view of a particular service.
- In web applications, Container will create a separate Servlet-Config object for each and every service.
- If we keep data in ServletConfig object then that data is shared to the respective service only.
- In web applications, Container will create ServletConfig object immediately after service instantiation and just before calling init() method in service Initialization.
- In web applications, Container will destroy ServletConfig object just before performing Service Deinstantiations.
- The Scope of the ServletConfig object is upto a particular service.
- The lifetime of ServletConfig object almost all same as the respective Service object lifetime.
- ServletConfig object is able to store parameters data, not attributes data.

Note:- Parameters data is the data available in the form of key-value pairs provided by html forms or web.xml file and which are stored in the objects at the time of creation. Attributes data is the data provided in the form of key-value pairs by the service at runtime in the objects after creation.]

Note:- Parameters data is treated of static data and attributes data is treated of dynamic data.]

To represent `ServletConfig` object in web applications, Servlet API has provided a predefined interface in the form of "`javax.servlet.ServletConfig`" and whose implementation classes are provided by all the server vendors.

In web applications, there are two ways to get `ServletConfig` object.

## ① By Overriding `init()` method in `Servlet` class

```
public class MyServlet extends HttpServlet
```

```
{  
    ServletConfig config;  
    public void init(ServletConfig config) throws ServletException  
    {  
        this.config = config;  
    }  
}
```

## ② By using `getServletConfig()` method from `Servlet`

```
ServletConfig config = getServletConfig();
```

If we want to get logical name of the `Servlet` from `ServletConfig` object then we have to use the following method.

```
public String getServletName()
```

If we want to manage initialization parameters in `ServletConfig` object, first we have to declare initialization parameters in `web.xml` file. To declare initialization parameters in `web.xml` file we have to use `<init-param>`

(91)

following tags.

<web-app>

---

<servlet>

---

<init-param>

<param-name> name</

<param-value> value</

</init-param>

---

</servlet>

---

</web-app>

---

→ If we declare initialization parameters like above  
 the Container will store all the specified initialization  
 parameters in ServletConfig object at the time of creation.

→ To get the value of a particular initialization parameter in  
 ServletConfig Object we have to use the following method.

public String getInitParameter(String name)

→ To get names of the initialization parameters from ServletConfig  
 object we have to use the following method.

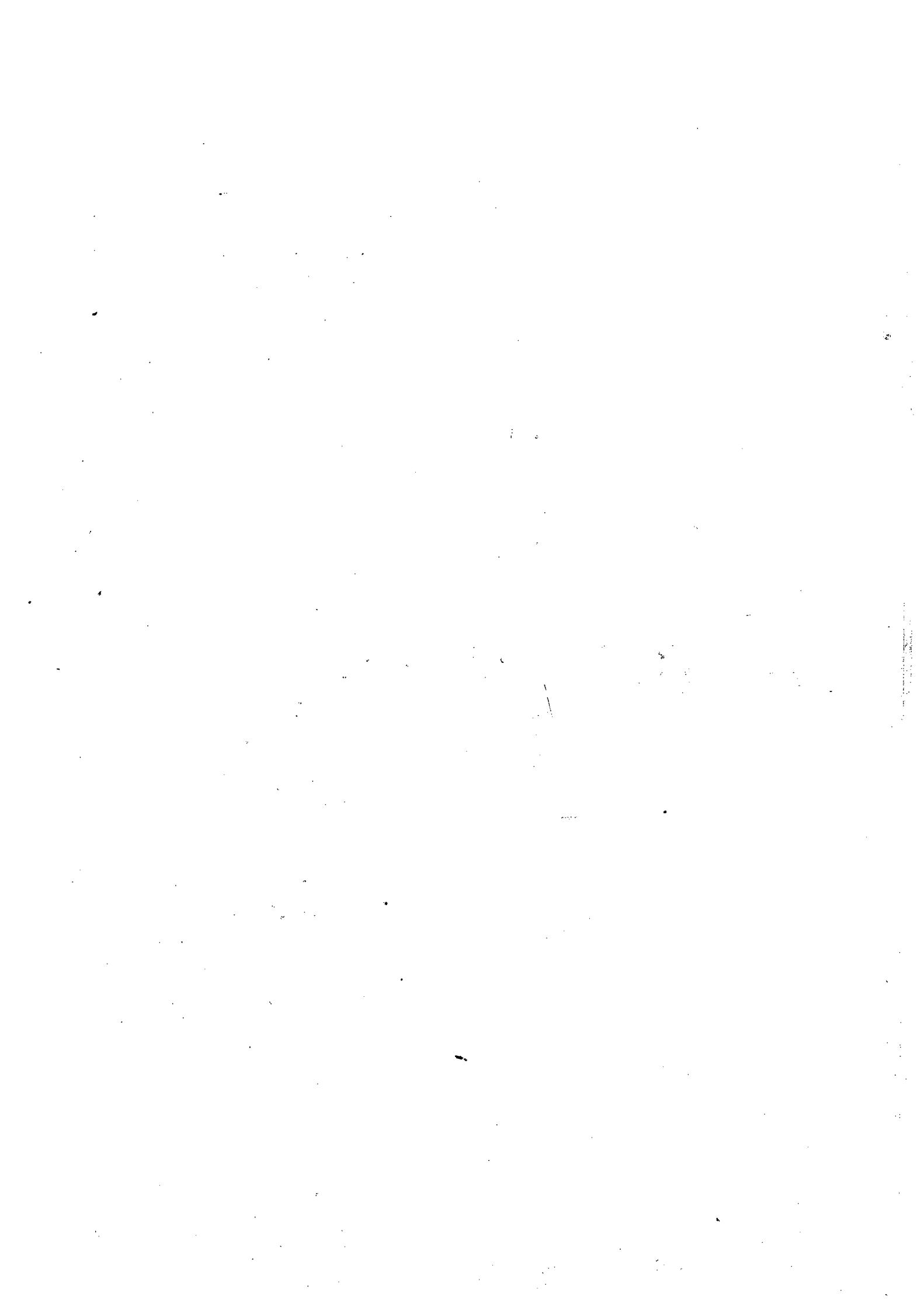
public Enumeration getInitParameterNames()

Eg:-

Enumeration e = config.getInitParameterNames();  
 while(e.hasMoreElements())

{  
 out.println(e.nextElement());

---



## Servlet Context:-

- \* \* Q: What are the differences between ServletConfig and ServletContext?
  - ServletConfig is an object, it able to provide all the configuration details of a particular servlet which we specified in web.xml file, which includes logical name of the servlet, initialization parameters,...
  - ServletContext is an object, it able to provide all the context details of a particular web application which we specified in web.xml file, which includes logical name of the web application and context parameters,...
  - ServletConfig object will provide the complete view of a particular servlet.
  - ServletContext object will provide the complete view of a particular web application.
  - In Web applications, Container will create a <sup>separate</sup> ServletConfig object for each and every servlet.
  - In server Space, Container will create a separate ServletContext object for each and every web application which we deployed.
  - In web-applications Container will create ServletConfig object immediately after servlet Instantiation and just before calling init() method in servlet Initialization.
  - In web applications, Container will create ServletContext object the moment when we start server or the moment when we deploy the web application.
  - In web applications, Container will destroy ServletConfig object just before performing servlet Deinstantiation.
  - In web applications, Container will destroy ServletContext object the moment when we shutdown the Server or at the time when we undeploy the web application.

6

→ The lifetime of ServletConfig object is the lifetime of the Servlet Object.

The lifetime of the ServletContext object is the lifetime of the respective web application in the server space.

→ If we keep data in ServletConfig object then that data is created upto the respective servlet.

If we keep data in ServletContext object then that data is stored upto throughout the respective web application.

→ The scope of ServletConfig object is upto a particular servlet.

The scope of ServletContext object is upto a particular Web application.

→ ServletConfig objects will provide less sharability.

ServletContext objects will provide more sharability.

→ Container will create ServletConfig object after getting first request from client except load on startup configuration.

Container will create ServletContext object at the time of Server Startup irrespective of getting first request from Client.

→ ServletConfig object is able to allow only parameters data that is static data.

cont...

(4)

ServletContext object is able to store both parameters data and attributes data that is both static data and dynamic data.

To represent ServletContext object in web applications, Servlet API has provided a predefined interface in the form of javax.servlet.ServletContext but its implementation classes are provided by all the server vendors.  
In web applications, there are three ways to get ServletContext object.

① By using getServletContext() method from ServletRequest :-

```
ServletContext context = request.getServletContext();
```

② By using getServletContext() method from ServletConfig :-

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

③ By using getServletContext() method from GenericServlet :-

```
ServletContext context = getServletContext();
```

If we want to get logical name of the web application, first we have to define logical name in web.xml file, for this we have to use the following tags.

```
<web-app>
```

```
-----<display-name>logical-Name</display-name>
```

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

3

→ To get logical name of the web application, we have to use the following method from `ServletContext`.

```
public String getServletContextName()
```

→ If we want to manage Context parameters in `ServletContext` object, first we have to declare them in web.xml file, for this, we have to use the following tags.

```
<Web-app>
```

```
  <Context-param>
```

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

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

```
  </Context-param>
```

```
</Web-app>
```

→ If we provide Context parameters and web.xml file like above then Container will store them in `ServletContext` object at the time of creation that is at the time of Server Startup or at the time of web application deployment.

→ If we want to get the value of a particular Context parameter from `ServletContext` object then we have to use the following method.

```
public String getInitParameter(String name)
```

E

→ If we want to get names of the Context parameters from ServletContext object then we have to use the following method.

(44)

```
public Enumeration getInitParameterNames()
```

→ In web applications, ServletContext object is able to allow both parameters' data and attributes data.

→ To set an attribute in ServletContext object we have to use the following method.

```
public void setAttribute(String name, Object value)
```

→ To get a particular attribute value from ServletContext object we have to use the following method.

```
public Object getAttribute(String name)
```

→ To remove an attribute from ServletContext Object we have to use the following method.

```
public void removeAttribute(String name)
```

→ To get names of the attributes from ServletContext object we have to use the following method.

```
public Enumeration getAttributeNames()
```



Q. What is Foreign Context?

→ Foreign Context is a ServletContext object of another web application being executed in the same server.

To get ForeignContext object we have to use the following method.

```
public ServletContext getServletContext(String path)
```

2

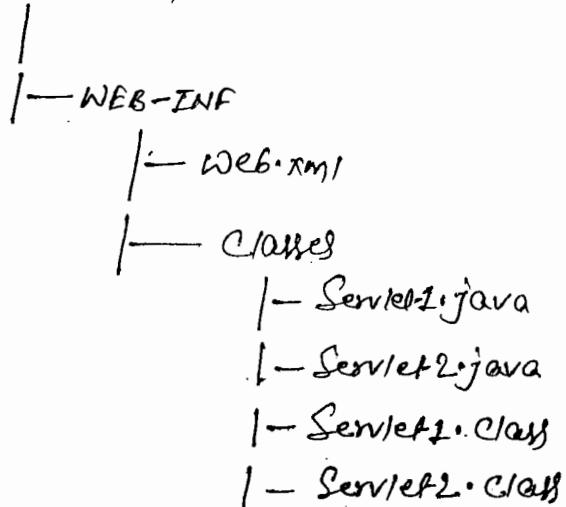
String → object → does casting.

14/12/2015

Note:- Supporting ForeignContext or not is completely depending on the server which we used. If the server is not supporting ForeignContext then getContext() method will return null value.

Q Consider the following web application.

Contextapp2



Ans ⇒ WEB.XML

```
<web-app>
  <context-param>
    <param-name>a</param-name>
    <param-value>AAA</param-value>
  </context-param>
  <context-param>
    <param-name>b</param-name>
    <param-value>BBB</param-value>
  </context-param>
  < servlet>
    < servlet-name>S1</servlet-name>
    < servlet-class>Servlet1</servlet-class>
```

(45)

```
<init-param>
  <param-name> C </param-name>
  <param-value> CCC </param-value>
</init-param>
<init-param>
  <param-name> D </param-name>
  <param-value> DDD </param-value>
</init-param>
</servlet>
< servlet-mapping>
  < servlet-name> S1 </servlet-name>
  < url-pattern> /first </url-pattern>
</servlet-mapping>
< servlet>
  < servlet-name> S2 </servlet-name>
  < servlet-class> Servlet2 </servlet-class>
  <init-param>
    <param-name> E </param-name>
    <param-value> EEE </param-value>
  </init-param>
  <init-param>
    &lt;param-name> F </param-name>
    <param-value> FFF </param-value>
  </init-param>
</servlet>
< servlet-mapping>
  < servlet-name> S2 </servlet-name>
  < url-pattern> /second </url-pattern>
</servlet-mapping>
</web-app>
```

W

## Ex:- Servlet1.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Servlet1 extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse
                       response) throws ServletException, IOException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        ServletConfig cfg = getServletConfig();
        ServletContext ctx = getServletContext();
        out.println("<html><body><h2>");
        Out.println("a → " + ctx.getInitParameter("a") + "<br>");
        Out.println("b → " + ctx.getInitParameter("b") + "<br>");
        Out.println("c → " + cfg.getInitParameter("c") + "<br>");
        Out.println("d → " + cfg.getInitParameter("d") + "<br>");
        Out.println("e → " + cfg.getInitParameter("e") + "<br>");
        Out.println("f → " + cfg.getInitParameter("f") + "<br>");
        Out.println("</h2></body></html>");
    }
}
```

## Ex:- Servlet2.java:-

(46)

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

public class Servlet2 extends HttpServlet.

[Note:- doGet() method is same as Servlet1.java doGet() method.]

- i) If we use the following request then what would be the response?

HTTP://localhost:1010/Contextapp2/first

O/P  
a → AAA  
b → BBB  
c → CCC  
d → DDD  
e → null  
f → null

- ii) If we use the following request then what would be the response?

HTTP://localhost:1010/Contextapp2/second

O/P  
a → AAA  
b → BBB  
c → null  
d → null  
e → EEE  
f → FFF

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## User Interface in web applications :-

15/12/2015-

& 16/12/2015

The main intention of user interface is, to

- ① To improve look and feel for the web applications.
- ② To provide entry point to the user in order to interact with server side application.
- ③ To provide environment to accept data from users.
- ④ To perform client side data validations with java script functions.
- ⑤ To specify different types of request from client to server.

To prepare user interface in web applications we have to use presentation logic, for this we have to use mainly HTML technology.

There are two types of user interfaces.

### ① Informational User interface:-

It able to display information at client browser like States of the server side actions like login success, login failure,... And displaying a particular database table data.

### ② Form-Based User interfaces:-

It able to include an HTML form to collect data from users and to submit that data to server side application.

→ In web applications, there are two ways to get User forms.

#### ① Static form Generation

#### ② Dynamic form Generation

(47)

## ① Static form Generation:-

In this approach, we have to prepare an HTML file with form logic under application folder, and we have to access it from client.

Ex:- loginform.html, registrationform.html, ...

## ② Dynamic form Generation:-

In this approach we have to prepare HTML form logic in a Servlet under classes folder and we have to access that service from client when we required form.

Ex:- EditformServlet, ...

In both static form and dynamic form, if we submit request from client to server then request data will be stored in Request Object.

In web applications, Request Object is able to store the following three types of data.

- ① Headers Data.
- ② Parameters Data
- ③ Attributes Data

### ① Headers Data:-

These are the key-value pairs stored in request object at the time of creation and which are representing metadata of the client browser like name of the browser, locale of the browser, zipping formats which are supported by the browser, encoding mechanisms which are supported by the browser, ...

To get value of a particular request header we have to use the following method.

```
public String getHeader(String header-name)
```

To get all header names which are available in Request object we have to use the following method.

```
public Enumeration getHeaderNames()
```

## ② Parameters Data:-

There are the key-value pairs stored in request object at the time of creation representing the data provided by the users at client browser in user forms.

To get value of particular request parameter we have to use the following method.

```
public String getParameter(String name)
```

To get more than one value of a particular request parameter we have to use the following method.

```
public String[] getParameterValues(String name)
```

To get all parameter names from request object we have to use the following method.

```
public Enumeration getParameterNames()
```

## ③ Attributed Data:-

These are the key-value pairs provided to the request object after creation and which are provided by the services at runtime.

→ To set an attribute in request object we have to use the following method. (98)

public void setAttribute(String name, Object value)

→ To get value of a particular attribute we have to use the following method.

public Object getAttribute(String name).

To remove an attribute from request object we have to use the following method.

public void removeAttribute(String name)

To get all attribute names from request object we have to use the following methods.

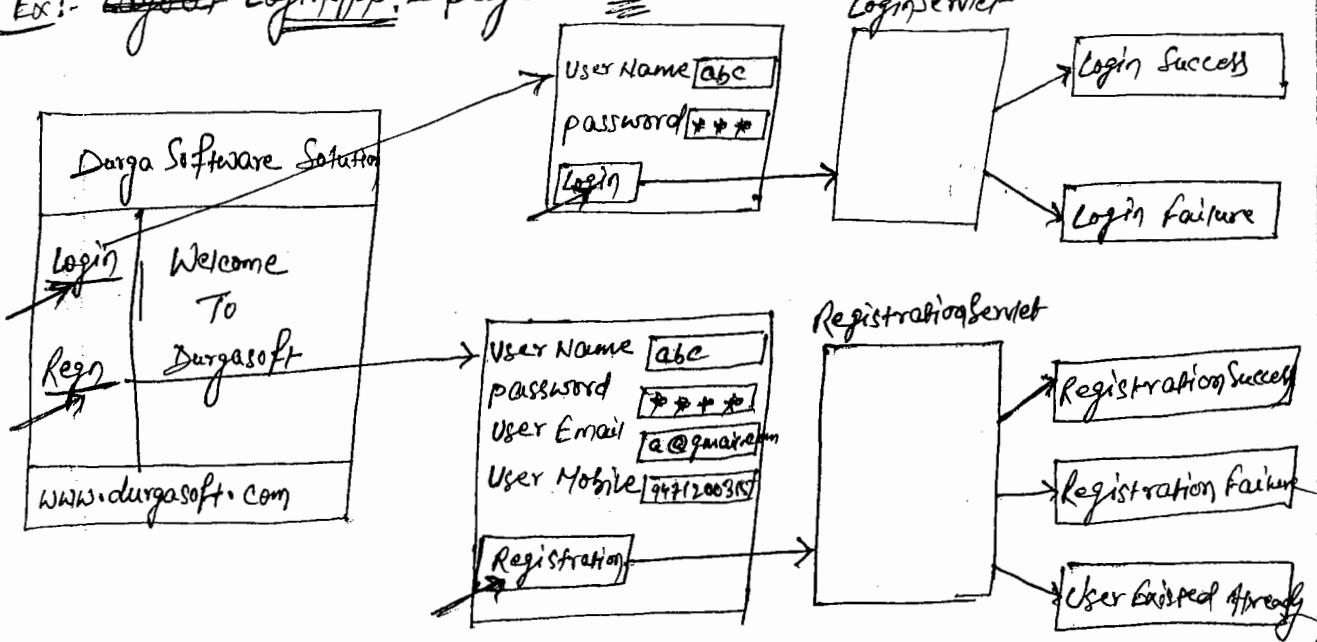
public EnumerationgetAttributeNames()

E

Ex:- Applications is of header (headersApp) → page-44 (Servlet)

Application is of Parameters (ParametersApp) → page-46 (Servlet) 17/12/15 18/12/15

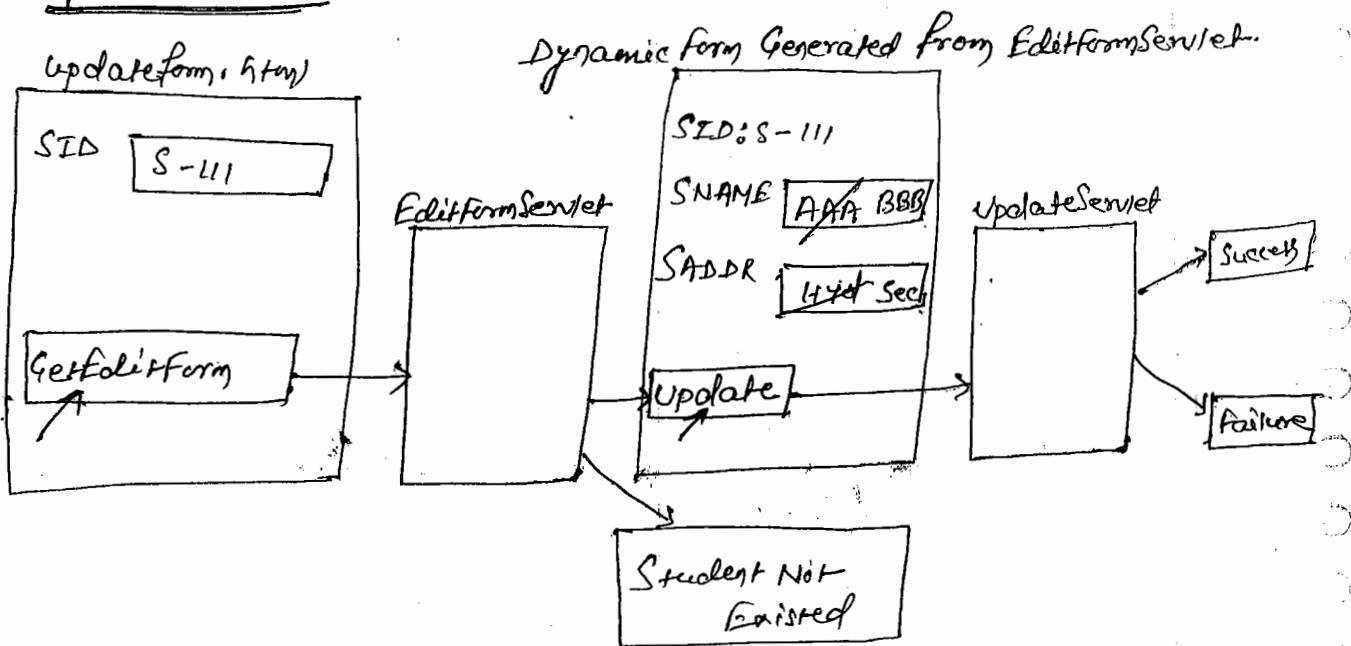
Ex:- Layout LoginApp:- page-52



Dynamicformapp :- page - 64

17/12/15  
18/12/15

Ex:- updateform.html



19/12/15

### Welcome files:-

→ Welcome file is a first page in web applications, it must be executed by the container automatically when we access the web application without specifying resource name in URL.

If we want to make any file as welcome file then we have to declare that file in web.xml file with the following tags.

<Web-App>

-----  
<Welcome-file-list>

<Welcome-file> file1 </Welcome-file>

<Welcome-file> file2 </

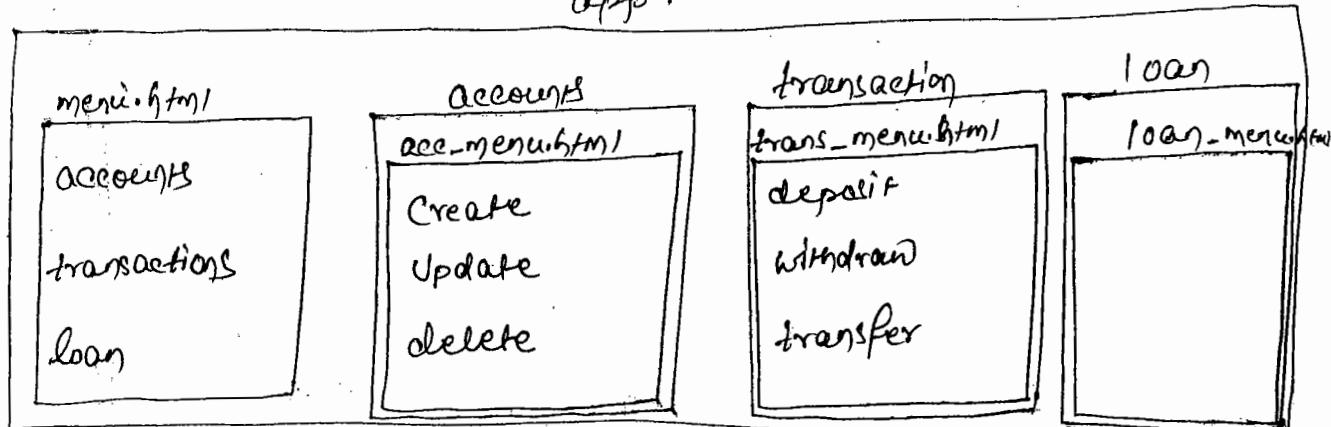
-----  
</Welcome-file-list>

</Web-App>

(49)

- In web applications, index.html and index.jsp are treated as default welcome files, which are not required welcome files configuration in web.xml file.
- From the above welcome files Configuration in web.xml file, we are able to provide more than one welcome file in a single web application but with respect to the module.
- In the above context, if we submit a request to a particular module then Container will take welcome file names from web.xml file, Container will goto module respective folder and Container will search for the welcome files as per the welcome files names declared in web.xml file.

app.



&lt;web-app&gt;

&lt;w-f&gt;

&lt;w-f&gt; menu.html&lt;/w-f&gt;

&lt;w-f&gt; acc-menu.html&lt;/w-f&gt;

&lt;w-f&gt; trans-menu.html&lt;/w-f&gt;

&lt;w-f&gt; loan-menu.html&lt;/w-f&gt;

http://localhost:1010/app/menu.html ✓

http://localhost:1010/app/accounts/acc-menu.html  
acc-menu.html ✓http://localhost:1010/app/transaction/trans-menu.html  
acc-menu.html trans-menu.html ✓http://localhost:1010/app/loan/loan-menu.html  
acc-menu.html trans-menu.html  
loan-menu.html ✓

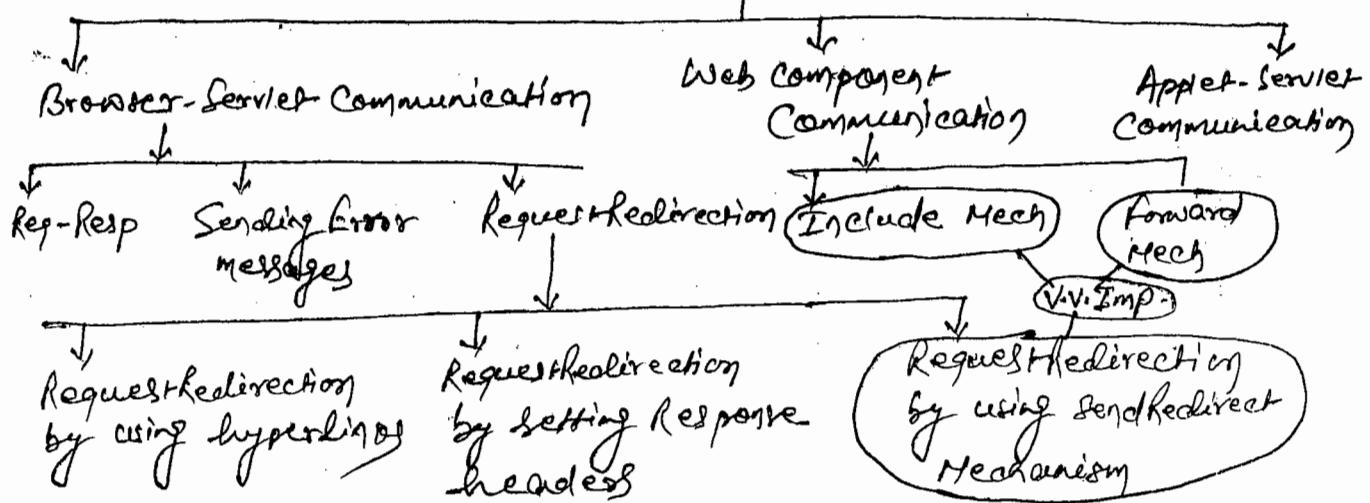
Note: In web applications, if we provide default welcome file [index.htm]/[index.jsp] and an explicit welcome file like abc.htm then container will execute explicit welcome file that is abc.htm. In web applications, when explicit welcome files are not configured in web.xml then only container will search for default welcome files.

## Servlet Communication:-

→ In web application development, it is not suggestable to provide the complete application logic within a single servlet, it is suggestable to provide application logic in more than one server side component. In this context, to process the request we must provide file communication between server side components, for this we have to use "Servlet Communication".

In web application we are able to provide service communication in the following three ways.

### Servlet Communication



→ In web application, Browser is acting as Client, it has to submit request to a servlet, Where servlet has to generate dynamic response to the respective client, here there must be a communication between Browser and servlet, it will come under Browser-Servlet Communication.

## ② Sending Error Messages:-

→ In web applications execution, if we have any mistake in Servlets execution or in JSPs execution then Container will display error messages at client Browser in Container defined format.

In Web Applications, if we want to display application specified error message at client browser in Container defined format then we have to use the following method.

```
public void sendError(int sc, String description)
```

Ex:-

```
response.sendError(505, "U R NOT eligible for this Recruitment process");
```

≡.

Ex:- Application Name :- SendErrorApp  
page :- 92

20/12/15

### Request Redirection:-

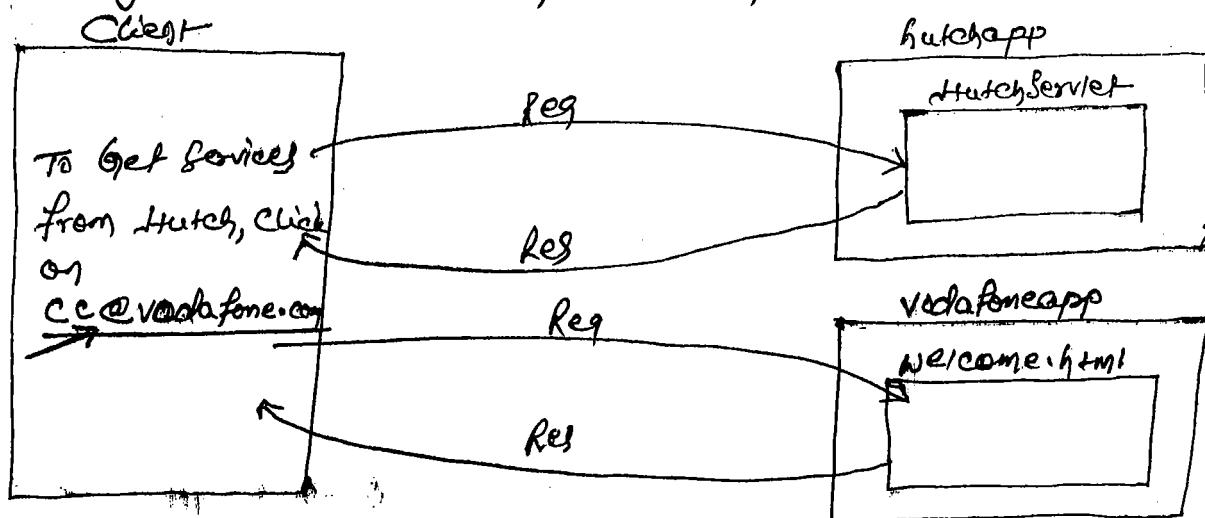
The process of passing request from one web application to another web application is called as RequestRedirection.

There are three approaches to perform Request Redirection.

- ① RequestRedirection by using hyper links.
- ② RequestRedirection by setting Response headers
- ③ RequestRedirection by using SendRedirect Mechanisms.

#### ① RequestRedirection by using hyperlinks:-

- ① Submit request to first web application
- ② First web application has to send an hyperlink as response, where the generated hyperlink must refer second web application.
- ③ We must click on generated hyperlink.
- ④ With the above step, new request will be send to second web application, where second web application will generate the required response.



## Hutchapp

```
|→ WEB-INF  
|→ web.xml  
|→ classes  
|→ HutchServlet.java  
|→ HutchServlet.class
```

## Vodafoneapp

```
|→ welcome.htm  
|→ WEB-INF  
|→ classes.
```

### Ex:- HutchServlet.java

```
import java.io.*;  
import javax.servlet.*;  
import javax.servlet.http.*;  
public class HutchServlet extends HttpServlet  
{  
protected void doGet(HttpServletRequest request,  
HttpServletResponse response) throws ServletException, IOException  
{  
response.setContentType("text/html");  
PrintWriter out = response.getWriter();  
out.println("<html>");  
out.println("<body>");  
out.println("<center>");  
out.println("<br><br>");  
out.println("<h2>");  
out.println("<To get services from Hutch<br>");  
out.println("<Click on<br>");  
out.println("<a href='http://localhost:1010/Vodafoneapp  
/welcome.html'>CC@vodafone.com</a>");
```

```
out.println("<h2><center></body></html>");
```

### Web.xml

```
<web-app>
  <service>
    <service-name> HutchServlet</service-name>
    <service-class> HutchServlet</service-class>
  </service>
  <service-mapping>
    <service-name> HutchServlet</service-name>
    <url-pattern> /Hutch </url-pattern>
  </service-mapping>
</web-app>
```

### Welcome.html (in vodafoneapp)-

```
<html>
  <body bcolor="red">
    <br><br>
    <font color="white" size="7">
      <b>
        powered by vodafone
      </b>
    <hr><hr><font><body></html>
  </font>
```

Note:- This approach is not giving guarantee to achieve RequestRedirection, because hyperlinks are not trustable in web environment.

## RequestRedirection by setting Response Header:-

- ① We have to submit request to first web application.
- ② At first web application, we have to set Redirectional static code to "Status Line Field" in response format and new web application url to "Location" response header in response format.
- ③ First web application will submit ~~response~~ to Client, where Client will recognize Redirectional status code from Status Line Field and identifying new web application from "Location" response header and Client will send another request to new web application as per the url.
- ④ Client will get response from new web application.

To set Redirectional Status Code to Status Line Field in Response

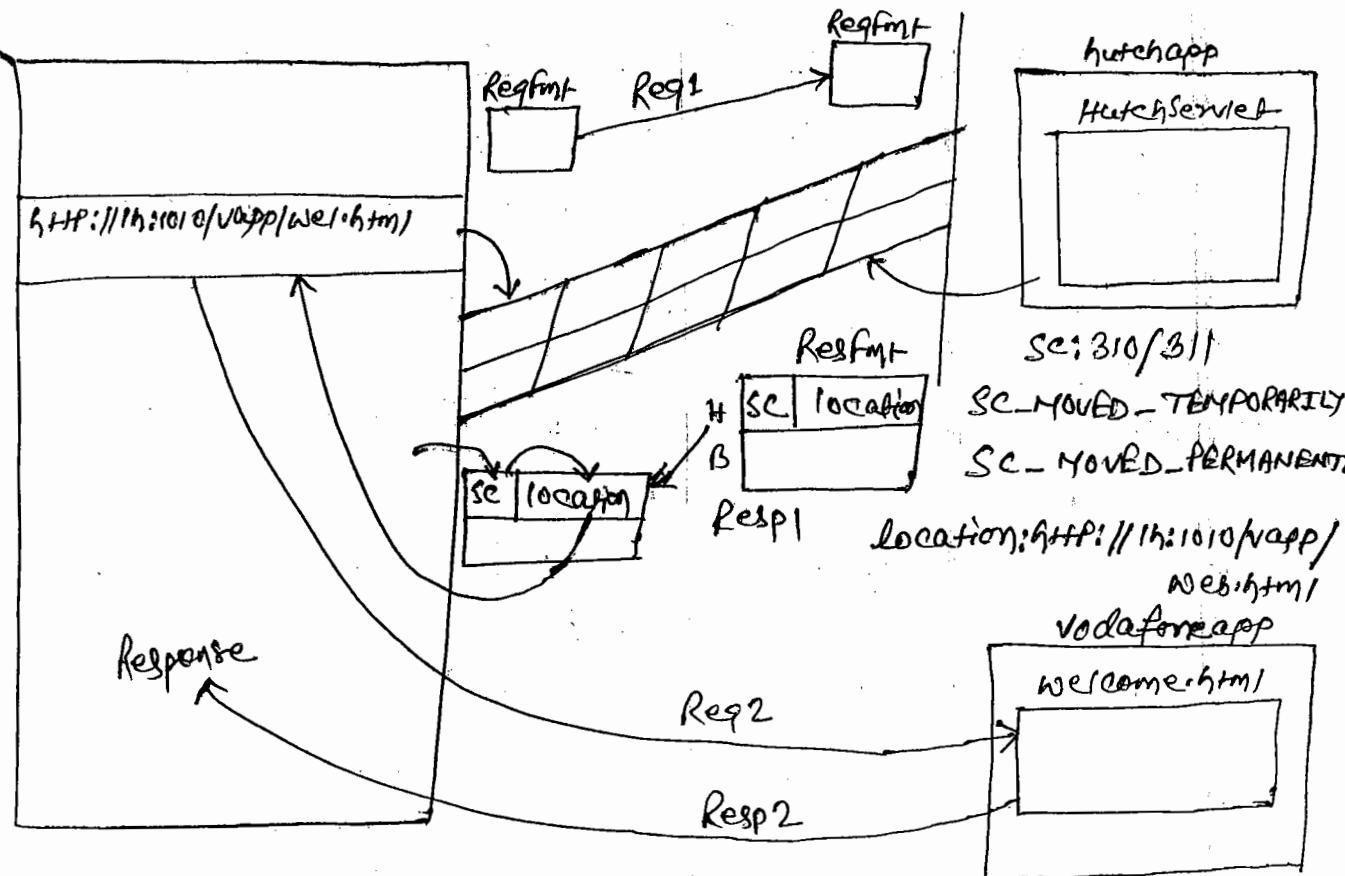
public void setStatus(~~int~~ SC)

Where SC may be SC-MOVED-TEMPORARILY or  
SC-MOVED-PERMANENTLY.

To set value to a particular response header we have to use the following method.

public void setHeader(String name, String value)

where "name" must be "location" and "value" must be new web application url.



Q:- HutchServlet.java :- (remaining are ~~is available in material~~ same as base ex.:-)

— import statements —

```
public class HutchServlet extends HttpServlet
```

```
{ public void doGet(--req, --res) throws SE, IOException
```

```
{ res.setStatus( HttpServletResponse.SC_MOVED_PERMANENTLY );
```

```
res.setHeader( "location", "http://localhost:1010/vodafone/welcome.html" );
```

```
}
```

```
=
```

Note:- In web appn, it is not suggestible to set values to the Response format explicitly.

## Request Redirection by Using sendRedirect Mechanism:-

→ In this approach, to implement Request Redirection we have to use the following method without setting status code value and location response header value in Response Format.

```
public void sendRedirect(String new-webapp-URL).
```

### Example:-

#### HutchServlet.java

```
public class HutchServlet extends HttpServlet  
{  
    public void doGet(--req, --resp) throws SE, IOE  
    {  
        res.sendRedirect("http://123:1010/vodafone/welcome.html");  
    }  
}
```

## Web Component Communication:-

→ The process of providing communication between more than one server side component is called as Web Component Communication.

→ In web applications, web component communication is possible in the following combinations.

Servlet - Servlet

Servlet - JSP

Servlet - HTML

JSP - JSP

JSP - Servlet

JSP - HTML

— — — —  
— — — —

→ In web applications, we are able to implement web Component Communication in the following two ways:

- (1) Include Mechanism
- (2) Forward Mechanism

The above two mechanisms are called as Request Dispatching mechanisms, because the above two mech. Must require RequestDispatcher object to implement web Component Communication.

RequestDispatcher is an object, it will provide very good environment either to include target resource response in the present resource response or to forward request from present resource to target resource.

→ To get RequestDispatcher object we have to use the following methods.

(1) From ServletContext:-

- (a) `getrequestDispatcher(-)`
- (b) `getnamedDispatcher(-)`

(2) From ServletRequest:-

- (a) `getrequestDispatcher(-)`.



Q. What is the difference between `getRequestDispatcher()` method and `getNamedDispatcher()` method?

→ Both the methods are used to get RequestDispatcher object.

To get RequestDispatcher object, if we use `getRequestDispatcher()` method then we have to pass the locator of the target resource as parameter.

-

(Note:- In the case of Servlet, locator is an URL pattern defined in web.xml file.)

→ To get RequestDispatcher object, if we used `getNamedDispatcher()` method then we have to pass the logical name of the target resource as parameter.

(Note:- In case of servlets, logical name is the name specified along with `<servlet-name>` tag in web.xml file.)

Q. What is the difference between `getRequestDispatcher()` method from `ServletContext` and `getRequestDispatcher()` method from `ServletRequest`?

→ `getRequestDispatcher()` method from `ServletContext` can be used to get RequestDispatcher object, but, we must pass relative path of the target resource as parameter.

→ `getRequestDispatcher()` method from `ServletRequest` can be used to get RequestDispatcher object, but, we have to pass either relative path of the target resource or absolute path of the target resource.

Note:-

→ To implement include and forward request dispatching mechanisms in web applications then we have to use the following method from javax.servlet.RequestDispatcher.

```
public void include(ServletRequest req, ServletResponse res)
    throws ServletException, IOException.
```

```
public void forward(ServletRequest req, ServletResponse res)
    throws ServletException, IOException.
```

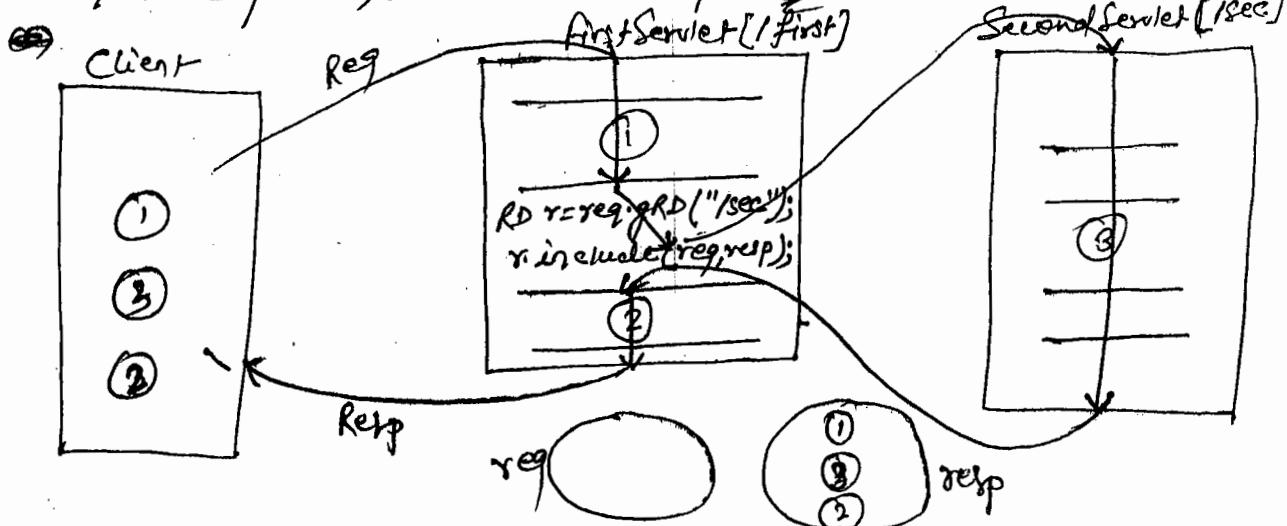
≡

22/12/2015

Q What is the difference between include and forward request dispatching mechanisms?

→ Include Mechanism:-

→ The main intention of include Request Dispatching mechanism is to include the target resource response in the present resource response.



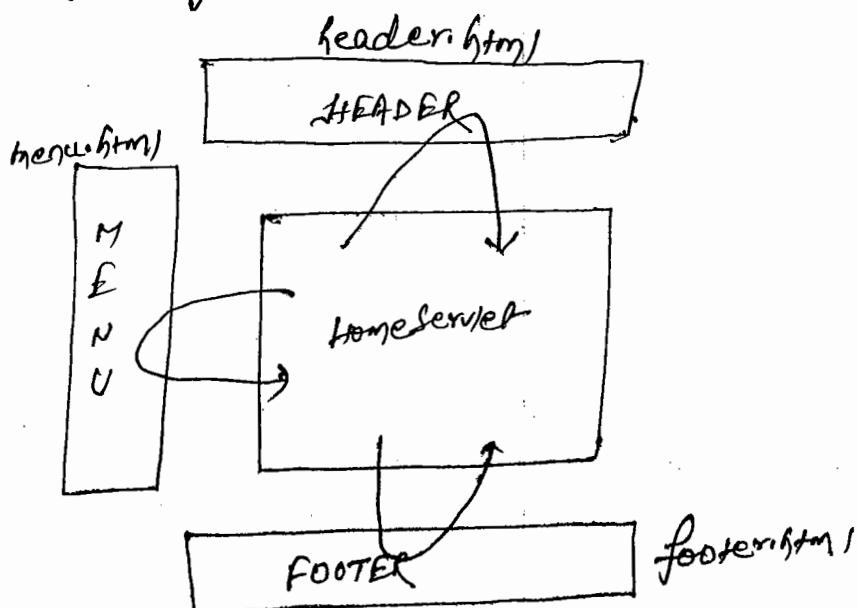
- When we submit request from client to first resource then request and response objects are created, where if Container executes any content in the first resource then response will be added to response object.
- In first resource when Container ~~executes~~ <sup>execute</sup> ~~include()~~ method then Container will send flow of execution to target resource with request and response objects.
- Note:- While sending response object to target resource the existed response is not deleted from response object.
- By the execution of the target resource some response will be added to the response object, at the end of the target resource Container will send flow of execution back to first resource without dispatching response to client.
- At first resource, Container will execute the remaining part, added some other response to response object, at the end of the first resource Container will dispatch overall response to client.
- In case of Include mechanism, client is able to ~~receive~~ <sup>receive</sup> all the resources response which are involved in the present request processing.
- In web app development, if we want to prepare web pages with a particular layout then we have to provide coding part of FOOTER, MENU, HEADER, BODY, ... at each and every web resource like services or JSPs, this approach will increase code redundancy. it is not suggestible to web apps.

→ To overcome the above problem, we have to use the following solution.

- (a) Prepare HEADER, MENU, FOOTER, ... separately in the form of HTML or JSP pages.

ex:- header.html, menu.html, footer.html, ...

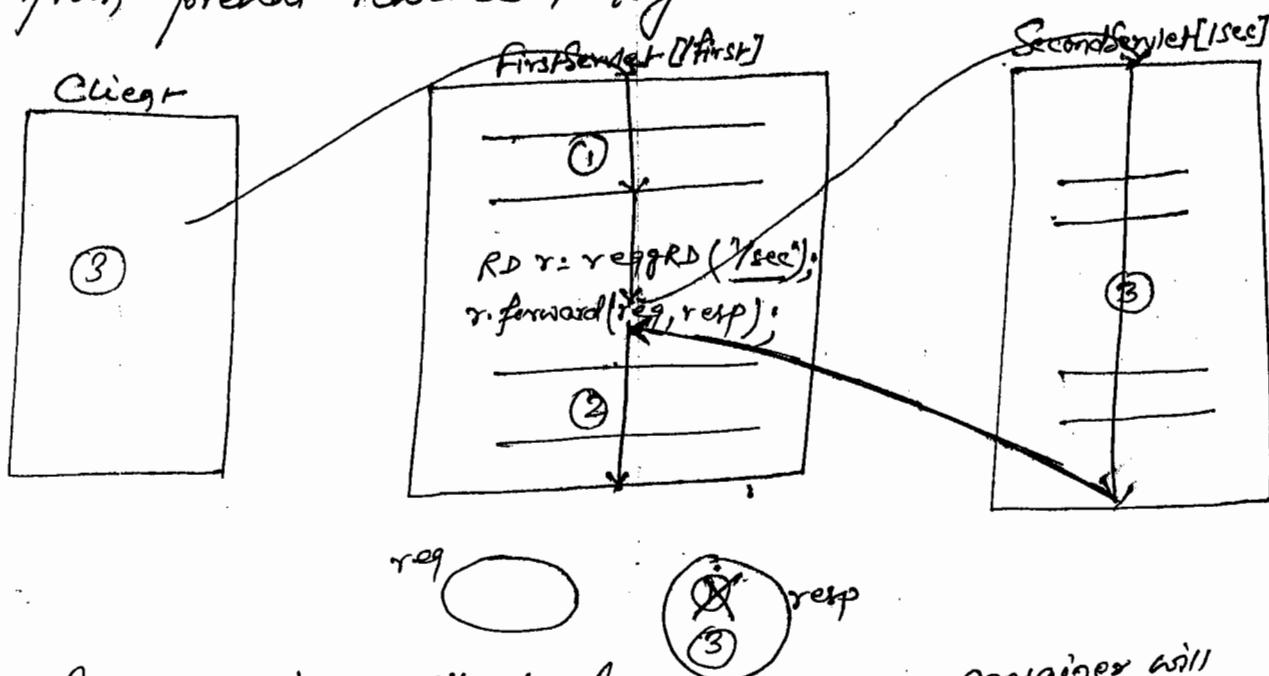
- (b) In which Servlet or JSP we need see above layout resources they include them in the respective Servlet or JSP by using "Include" Request Dispatching mechanisms.



## Forward Mechanism:-

23/12/2015

- The main intention of forward mech. is to forward request from present resource to target resource.

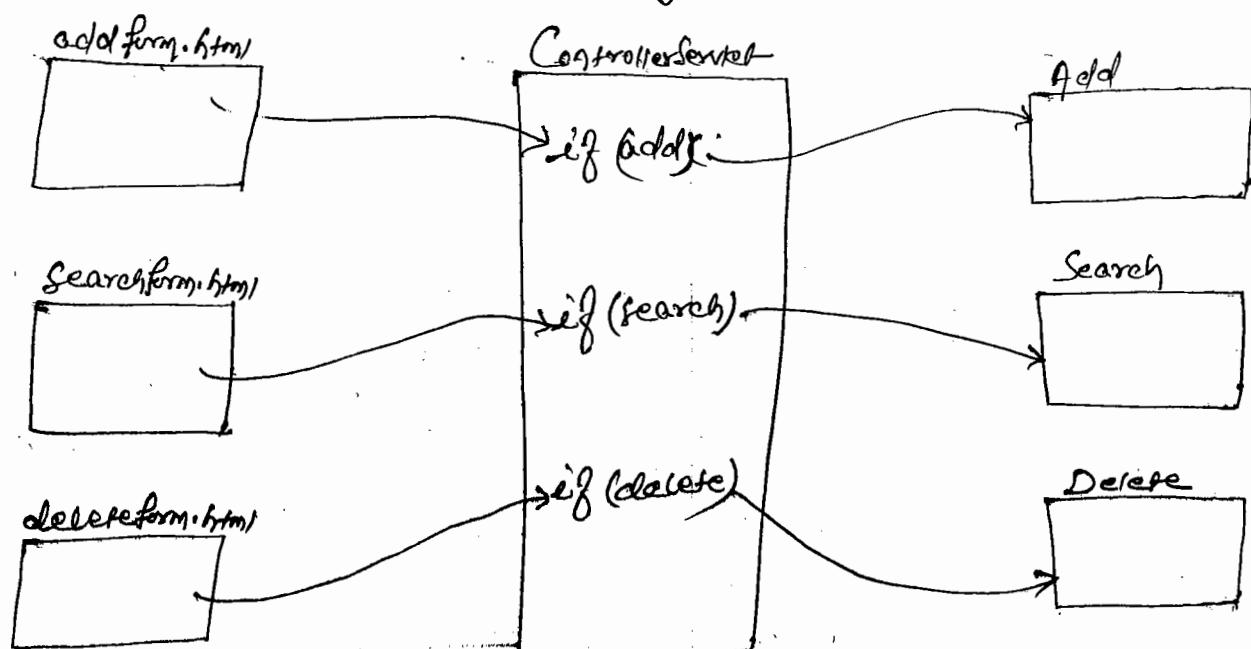


→ If we submit request to first resource then container will create request and response objects. If container executes any content in first resource then the respective response will be added to response object. When container encounters rd.forward() method then container will send request and response object to target resource along with flow of execution by deleting existed response in response object.

→ By the execution of the target resource, some response is added to response object, at the end of the target resource, container will dispatch overall response to client.

[Note:- At the end of the target resource container will dispatch response to client but flow of execution is coming back to first resource and execute the remaining part in first resource.]

- In case of forward mechanism, client is able to receive only target resource response.
- In MVC based web applications, we have to use a service or controller, it has to take all the requests, it has to identify to which module this request is coming then it has to forward request to the target module, to achieve this controller service functionality, we have to use "forward" request Dispatching mechanism.



≡

≡

Q. What is Servlet Chaining?

→ The process of including or executing more than one Servlet in a single request by using request dispatching mechanisms is called as "Servlet Chaining".

≡

Q. What is the difference between SendRedirect Mechanism and forward Mechanism?

→ SendRedirect is one of the mechanism to achieve Request Redirection, where it able to provide communication between two resources which may be available with in a single server or may be available in two different servers.

→ To achieve Request Redirection by using SendRedirect mechanism Client has to send two requests.

Forward is one of the Request Dispatching Mechanism, it able to provide communication between two resources which must be available with in a single server.

To achieve forward mechanism Client has to send only one request.

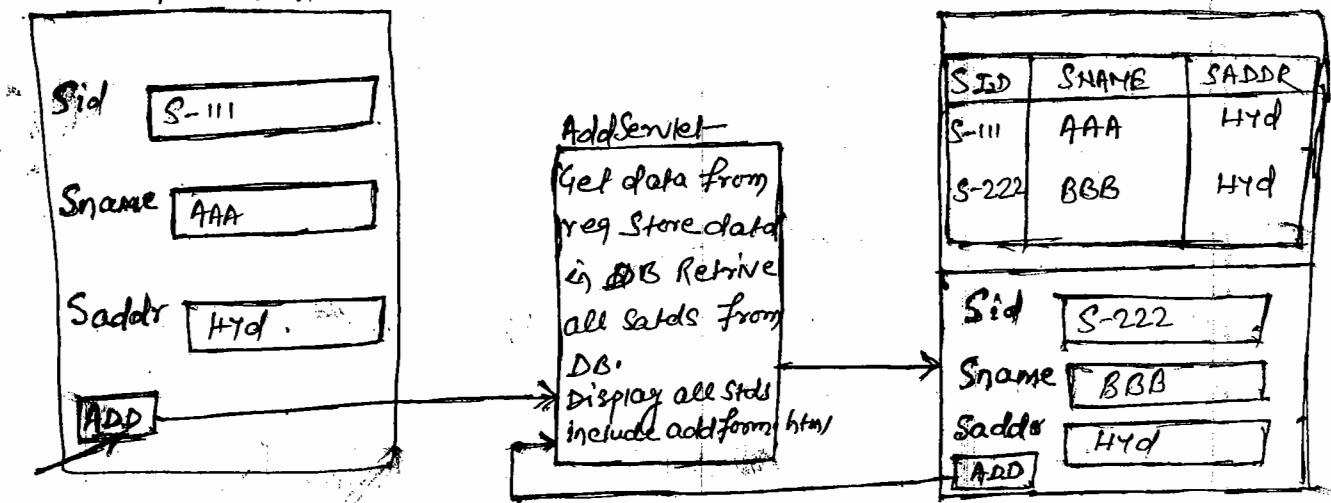
≡

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

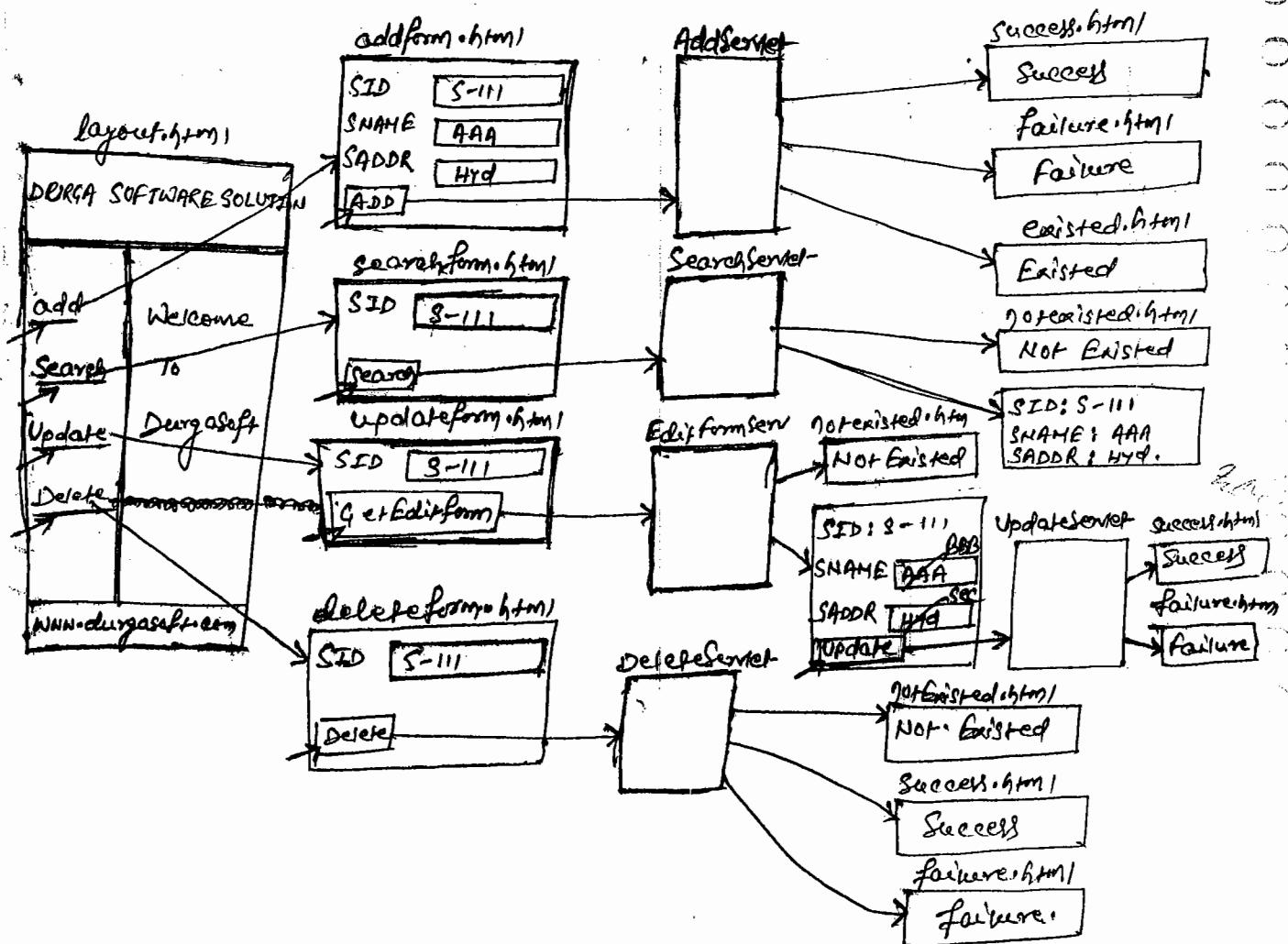
Ex:- includeapp:- page NO - 103 (product replaced to Student)

24/12/15

addform.htm



25/12/2015



25/12/2015

## Applet-Servlet Communication :-

→ In general in web applications, we will use a browser as Client, when we submit request from client browser then client browser will send request to protocol, Client browser will make protocol to establish connection with server as per the server IP address and server port no. and client browser will push request to protocol.

After executing server side application, protocol will transfer response to Client browser, where client browser is able to get response from protocol and it will display the response.

In web applications, if we are an Applet as Client then developer must take explicit responsibilities like preparing URLs, Establish connection between applet and server, Sending requests, ... explicitly.

In web applications, if we want to implement Applet-Servlet communication we have to use the following steps.

① Create java.net.URL object with an URL:-

```
URL u=new URL("http://localhost:1010/loginapp/login?username=  
aaa&pwd=abc123");
```

② Establish <sup>URL</sup> Connection with a particular URL :-

```
URLConnection uc=url.openConnection();
```

3

### ③ Enable request and response:-

```
uc.setDoInput(true);  
uc.setDoOutput(true);
```

### ④ Get InputStream from URLConnection:-

```
InputStream is = uc.getInputStream();
```

```
BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

### ⑤ get Response and Display Response:-

```
String response = br.readLine();  
System.out.println(response);
```

Note :- In JAVA/J2EE applications, AWT, APPLETS and SWING are providing very good look and feel when compare with all the other view related technologies, but ~~is~~ only the problem is, Applets will take more time to complete initialization, it will reduce application performance.

Ex:- page No-116 (material)

## Session Tracking Mechanisms:-

- In the web applications like Online shopping, Online business, e-banking, ... it is Convection to manage Clients previous requests data at the time of processing later requests.
- To achieve the above application requirement, we are unable to use request object, because in web applications Request object is created when request is coming to Container and request object is destroyed when response is reached to client.
- To achieve the above application requirement, we are able to use ServletContext object, but it unable to provide Separation between multiple users data at server side.
- In web application to manage clients previous request data at the time of processing later requests and to provide clear cut separation between multiple users data at Server Side we have to use a set of mechanisms at Server Side called as "Session Tracking mechanisms".
- Session is a fine duration, it will start at the starting point of the client conversation with server and it will terminate at the ending point of the client conversation with the server.
- During a particular Session, the data which is transferred between Client and server is called as "State of the Session".
- In web applications, a separate session will be created for each and every user, to keep track of all these sessions at Server we have to use "Session Tracking mechanisms".

≡

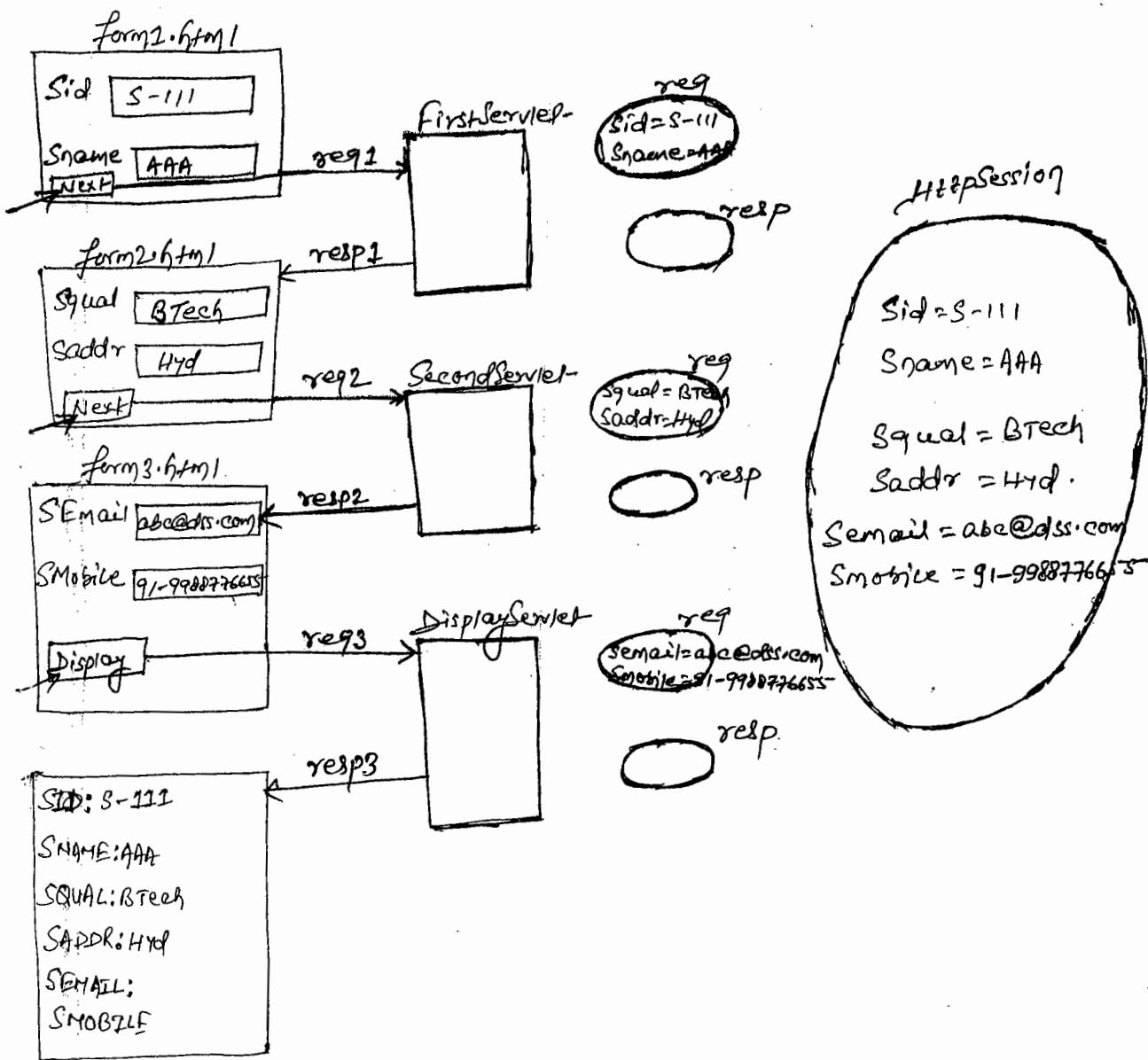
There are four tracking mechanisms in web applications.

- ① HttpSession Session Tracking mechanism.
- ② Cookies Session Tracking mechanism.
- ③ URL-rewriting Session Tracking mechanism.
- ④ Hidden form fields Session Tracking mechanism.

Note:- From the above session tracking mechanisms, Hidden form fields Session tracking Mechanism is not provided by Servlet API officially, it's purely Developers creation.

### ① HttpSession Session Tracking Mechanism :-

- HttpSession is an object provided by the container for each and every user, it able to store data in the form of attributes (as per the application's requirement).
- In HttpSession object, we have to perform the following actions at each and every request.
  - ① Get all request parameters from request object.
  - ② Get HttpSession object.
  - ③ Store all request parameters in HttpSession object.
  - ④ Forward request to next form.
  - ⑤ If the request is last request then get all the previous requests data from HttpSession object and use the data in web application as per the requirement.



- In web applications, to represent HttpSession object, Servlet API has provided a predefined interface in the form of javax.servlet.http.HttpSession and its implementation classes are provided by all the server vendors.
- In web applications, to get HttpSession object we have to use the following methods from HttpServletRequest.
  - (1) public HttpSession getSession()
  - (2) public HttpSession getSession(boolean b)

Q. What is the difference between getSession() method and  
getSession(false) method?

→ Both the methods are used to get HttpSession object. To get HttpSession object, if we use getSession() method, then container will check whether any HttpSession object is existed or not w.r.t. the present user, if any HttpSession object is existed then container will return that existed HttpSession object reference from getSession() method. If no HttpSession method is created w.r.t. the present user then container will create a new HttpSession object and container will return the generated HttpSession object reference from getSession() method.

→ To get HttpSession object - if we use getSession(false) method then Container will check whether any HttpSession object is existed or not w.r.t. the present user. If HttpSession object is existed w.r.t. the present user then container will return the existed HttpSession object reference. If no HttpSession object is existed w.r.t. the present user then container will return "null" value from getSession(false) method without creating new HttpSession object.

[Note:- getSession(true) method functionality is same as  
getSession() method functionality.]

→ If we want to destroy HttpSession object explicitly then we have to use the following method.

```
public void invalidate()
```

→ If we want to destroy HttpSession object after the specified idle time then we have to use the following method.

```
public void setMaxInactiveInterval(int time)
```

→ If we want to get idle time which is specified to HttpSession object then we have to use the following method.

```
public int getMaxInactiveInterval()
```

→ If we want to set an attribute in HttpSession object then we have to use the following method.

```
public void setAttribute(String name, Object value)
```

→ If we want to get a particular attribute value from HttpSession object then we have to use the following method.

```
public ObjectgetAttribute(String name)
```

→ If we want to remove an attribute from HttpSession object then we have to use the following method.

```
public void removeAttribute(String name)
```

E

→ If we want to get all the attribute names from HttpSession object then we have to use the following methods.

public EnumerationgetAttributeNames()

Q In web applications, Container has to manage a separate HttpSession object for each and every user. If multiple users are accessing web application then container has to manage multiple HttpSession objects. In this context, how it is possible for container to get user respective HttpSession object among the multiple HttpSession objects?

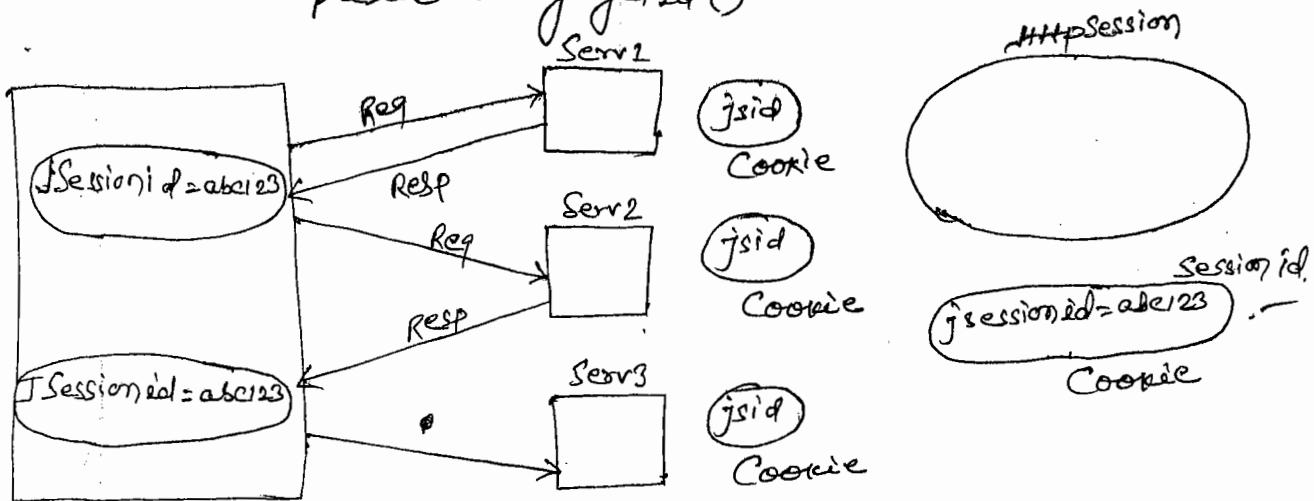
→ In web applications, when we create HttpSession object then Container will create an unique identity in the form of hexa decimal value that is "Session Id". When we create HttpSession object, Container will manage Session id value in the form of a cookie.

As per the basic nature of the cookies, Session id cookie will transfer from server to client along with response and from client to server along with request automatically.

→ In web apps, if we access getSession() method, Container will check any Session id cookie is available with request or not, if it is available then container will get Session Id cookie from request and Container will identify HttpSession Object on the basis of Session Id value.

→ To get Session Id value explicitly we have to use the following method.

```
public String getId()
```



### Drawbacks:-

- ① In HttpSession Session Tracking mechanism, Container will identify user respective HttpSession object by transferring session id Cookie between client and server. In this context, if we disable Cookies at client browser then it's not possible to get User respective HttpSession object at server, it will not execute HttpSession session tracking mechanism.
- ② In HttpSession Session tracking mechanism, Client's Conversation is maintained at server side in the form of HttpSession objects, here, if we increase more no. of users then more no. of HttpSession objects are created, it will increase burden to the server.

✓

## Cookies Session Tracking Mechanism:-

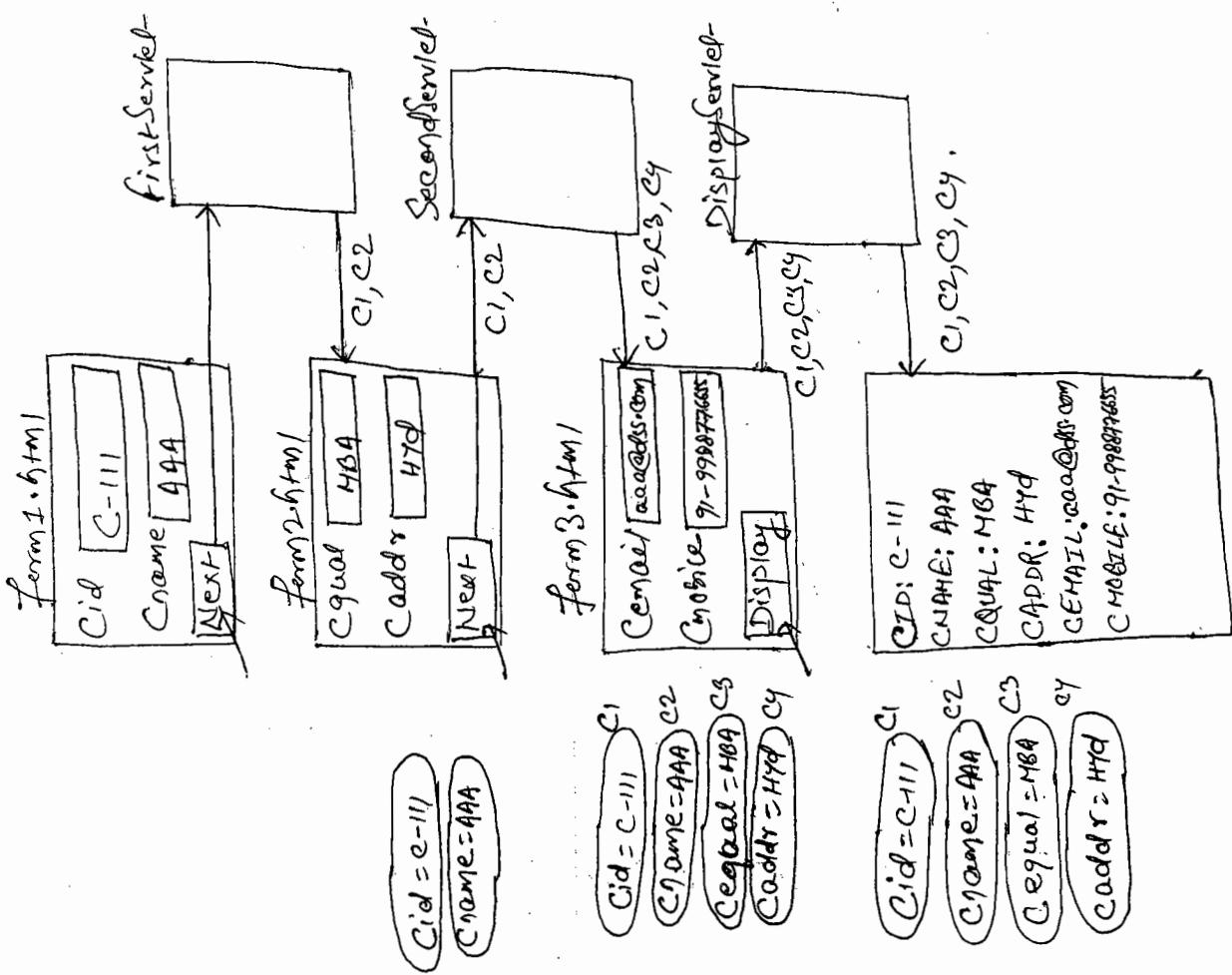
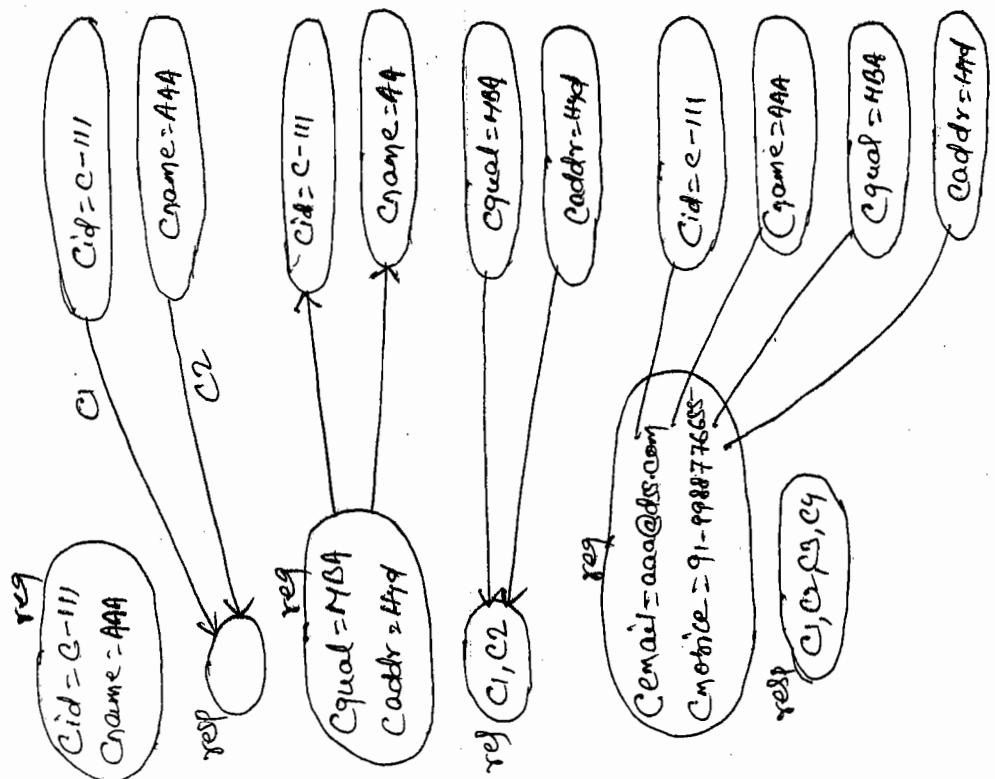
→ Cookie is a small object, it able to store a single name-value pair.

→ As per the basic nature, cookies are transferred from server to client along with response and from client to server along with request automatically.

In web applications, if we want to implement cookie session tracking mechanism then we have to perform the following steps at each and every request.

- ① Get request parameters from request object.
- ② Create a separate cookie for each and every request parameter.
- ③ If we create cookie first time then attach cookie to Response object. If any cookie is coming along with request then that cookie will be added to Response object automatically.
- ④ Forward request to next form.

→ When we submit response to client then the cookies which added to response object are transferred to client and stored at client machine. In this case, if we submit request from client to server then all the cookies which are stored in client machine will be transferred to server automatically.



→ In web applications to represent cookie object  
Servlet API has provided a predefined class in  
the form of "javax.servlet.http.Cookie".

To Create cookie class object we have to use the  
following constructor.

public Cookie(String name, String value)

Ex:- Cookie c1 = new Cookie("name", "AAA");

→ To add a cookie object to response, we have to  
use the following method.

public void addCookie(Cookie c)

Ex:- response.addCookie(c1);

→ To get all the cookie in the form of `Cookie[]`  
from request object we have to use the following  
method.

public Cookie[] getCookies()

Ex:- Cookie[] c = request.getCookies();

→ To get name and value from cookie object we have  
to use the following methods.

public String getName()

public String getValue()

Ex:- String name = c.getName();  
String val = c.getValue();

→ In web apps, we can provide comments also to the Cookies. To set a comment to the cookie and to get comment from the cookie we have to use the following methods.

public void setComment(String comment)

public String getComment()

→ For the cookies we are able to provide version numbers also, to set a particular version no. to the cookie and to get version no. from cookie we have to use the following methods.

public void setVersion(int no)

public int getVersion()

→ In web application, we can define lifetime to the cookies. To set age to the cookie and to get age of the cookie we have to use the following method.

public void setMaxAge(int time)

public int getMaxAge()

→ For cookies, we can set the location information where we want to store at client. To set path to the cookie and to get path from cookie we have to use the following methods.

public void setPath(String path)

public String getPath()



→ for cookies, we can set domain information also,  
to set domain from which website cookies are coming  
and to getDomain from Cookie we have to use  
the following methods.

```
public void setDomain(String domain)  
public String getDomain()
```

≡

26/12/2015

### Drawbacks:-

- ① In Cookies Session Tracking mechanism, user's data is stored at Client machine in the form of Cookies, it is open to every user of that machine, it will not provide security for the application's data or user data.
- ② In Cookies Session Tracking mechanism, Client's previous request data is transferred between Client and Server, in this case, if Client browser disable Cookies then Cookies Session Tracking mechanism will not be executed.  
To Overcome the above problems we have to use URL-rewriting Session Tracking mechanism.

≡

### ③ URL-Rewriting Session Tracking mechanism:-

26/12/2015

→ In both HttpSession session tracking mechanism and Cookies Session Tracking mechanism, if Client browser disables Cookies then both the mechanisms are not executed their functionalities. In case of Cookies Session Tracking mechanisms, app's data is stored at client machine, so it is not providing security for the app's data.

To Overcome all the above problems we have to use URL-Rewriting Session Tracking mechanism.

URL-Rewriting Session Tracking mechanism is same as HttpSession Session Tracking mechanism, but in URL-Rewriting Session Tracking mechanism we will use dynamic forms in order to append SessionId value to URL in the next form.

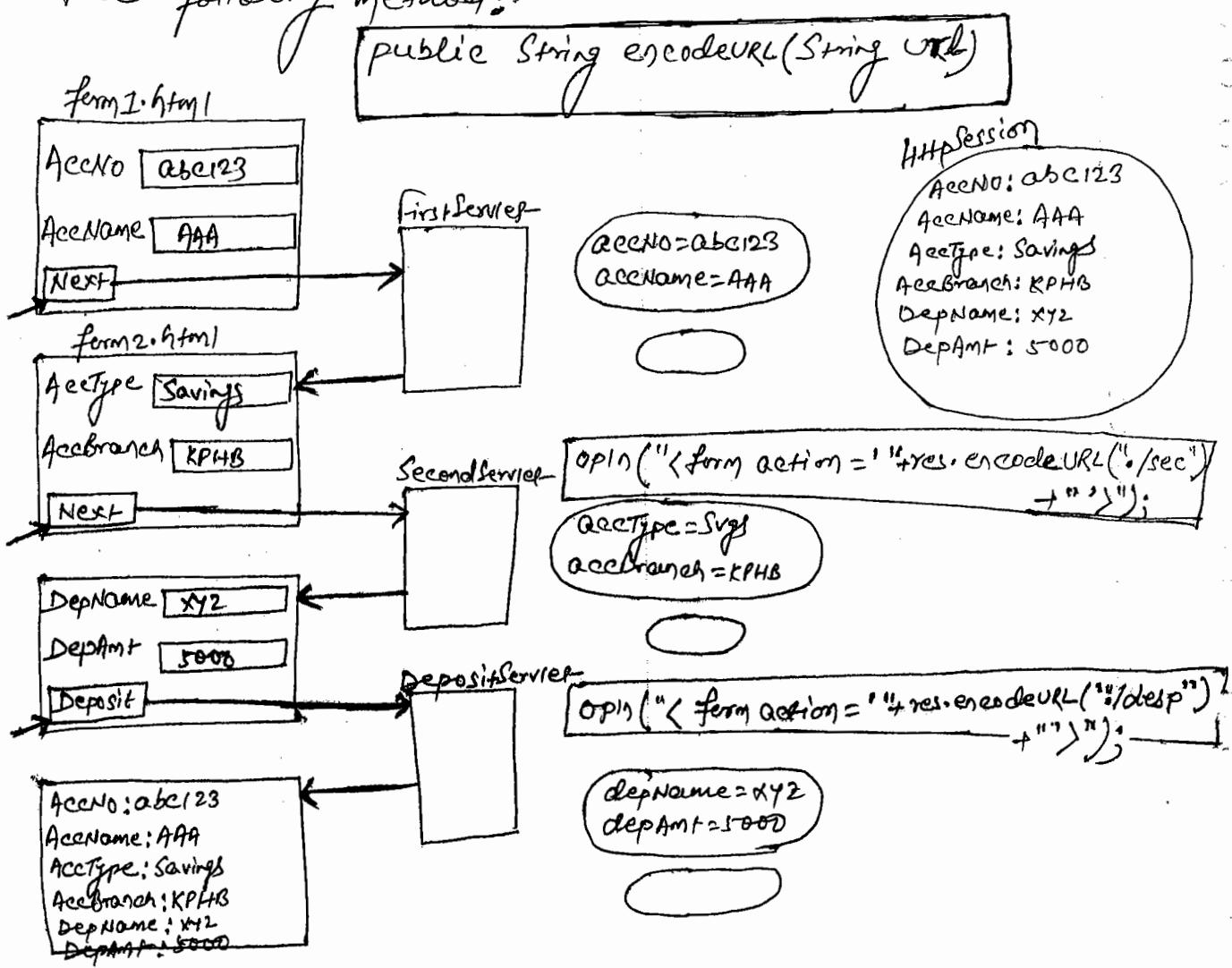
In HttpSession Tracking mechanism, we are depending on a Cookie to get SessionId value, but in URL-Rewriting Session Tracking mechanism, we are not depending on Cookie, we are Rewriting next form URL with SessionId value and we will get SessionId value from "SessionId" request header directly in the next request processing. In this Context, even Client browser disable Cookies, URL-Rewriting Session Tracking mechanism will execute its functionalities.



In web applications, if we want to implement URL-Rewriting mechanisms <sup>Session Tracking</sup> then we have to use the following steps at each and every request.

- ① Get all request parameters from request object.
- ② Get HttpSession object.
- ③ Store all request parameters in HttpSession object.
- ④ Generate dynamic form by rewriting URL with SessionID value.

To append SessionID value to the URL, we have to use the following method:



27/10/2015

### Drawbacks:-

- In URL Rewriting Session Tracking mechanism, we must prepare only dynamic forms, not static forms, because we have to rewrite url in the next form generation with Session\_id.

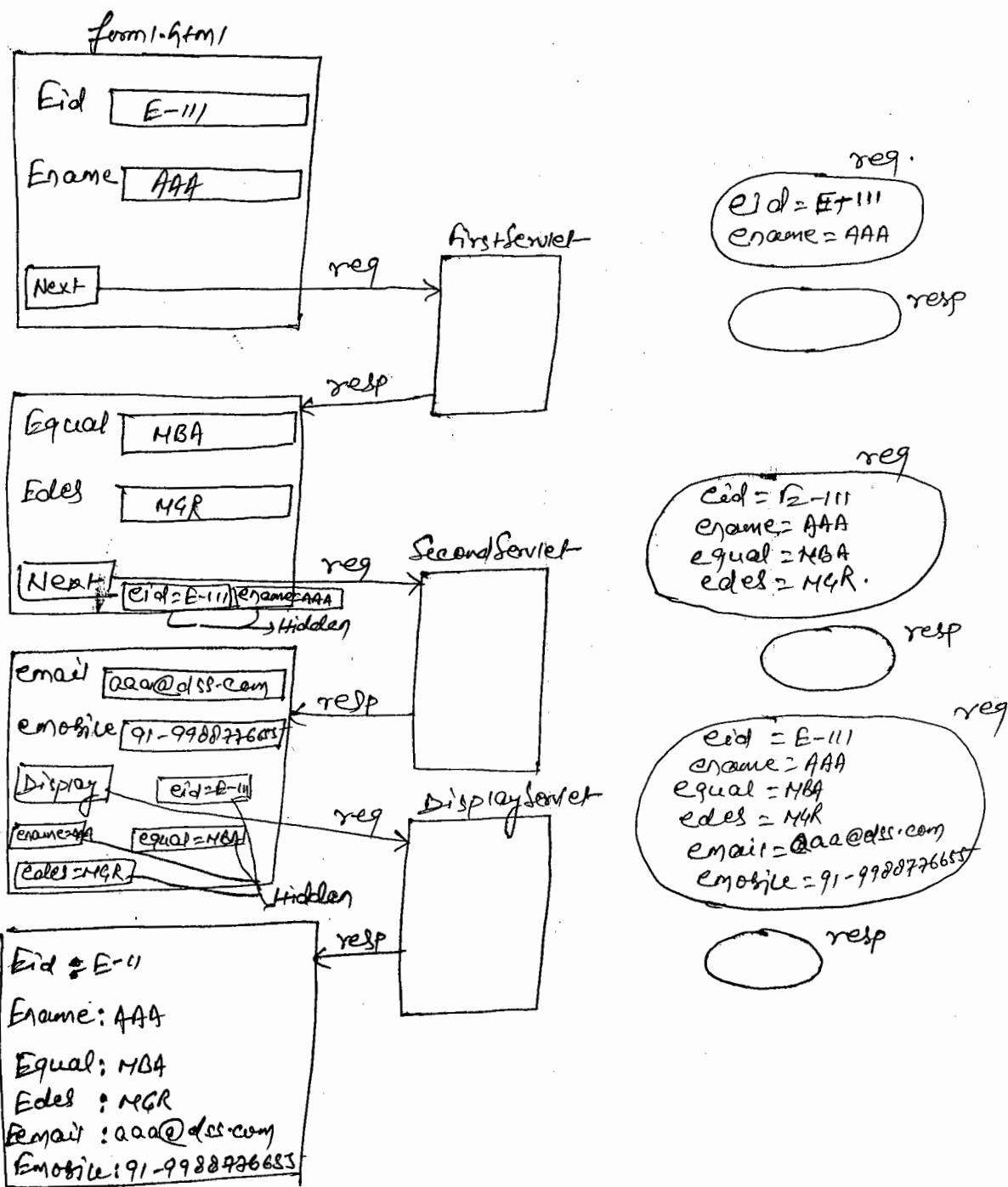
### ④ Hidden Form Fields Session Tracking Mechanism:-

27/12/2015

- This mechanism is not official mechanism provided by SUN Microsystems, it is purely developer's creation.
- If we want to implement this session tracking mechanism in web applications then we have to perform the following actions at each and every request.
  - a) Get all the request parameters from request object.
  - b) Prepare dynamic form, where we have to provide all the present request parameters in the form of hidden fields.
  - c) If we submit request from dynamic form [Next Form] then all visible field data and hidden field data will be send to server side as request parameters.

Note:- With the above mechanism we are able to manage client previous request data at the time of processing later request.

P.T.O



### Drawbacks:-

- ① It is not official mechanism from Sun Microsystems, so that, no developer will use this mechanism in their apps.
- ② At each and every request we have to prepare dynamic form as text form.

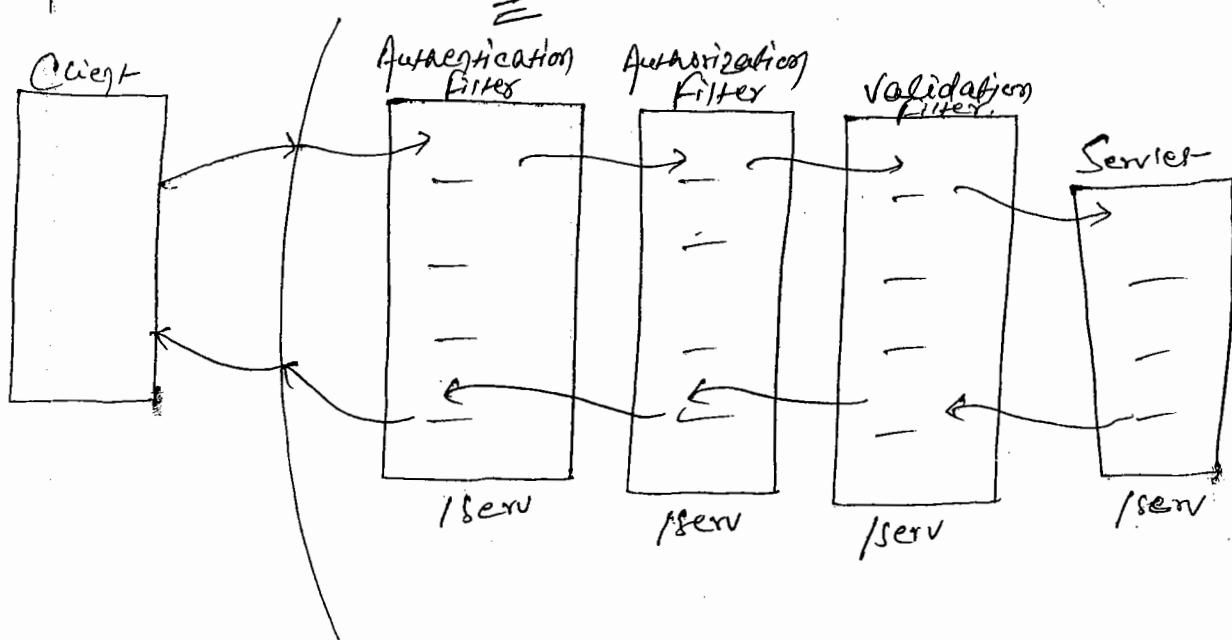
2F/12/15-

## Filters:-

→ In general, in web apps, it's convention to provide some pre-processing services and post-processing services like Authentication, Authorization, Data validations, Data Compression and Decompression, Data Encoding and Decoding, ... for the web resources like Servlets, JSPs, ... where business logic is existed.

To provide the above specified pre-processing and post-processing services, Sun Microsystems has provided a separable server-side component in the form of "filter" in Servlet API.

Filter is a Server Component, which must be executed by the Container automatically when we submit request to the respective Servlet or JSP.



→ When we submit request from Client to Server, Container will take request and Container will perform five following actions -

Cont...

- ① Container will go to web.xml file and identify the target service to which request is coming.
- ② Container will check whether any filter(s) is associated with the respective service or not.
- ③ If any filter is associated with service then Container will execute filter first, if all the constraints are satisfied in filter then container will forward request to next filter or to the service if no more filters are existed.
- ④ If filter constraints are not satisfied then container will generate some response to client without forwarding request to next filter or to the service.

→ If we want to use filters in web applications then we have to use the following steps:

- ① Create Filter class
- ② Configure filter class in web.xml file.
- ③ Create filter class:-

To create filter class we have to declare an user defined class, it must implement javax.servlet.Filter interface and it has to provide implementation for all the methods of javax.servlet.Filter interface.

S

```
public interface Filter {
    public void init(FilterConfig config) throws ServletException;
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain fc) throws ServletException, IOException;
    public void destroy();
}
```

public class Myfilter implements Filter  
— implementation for all the methods of filter interface —

→ Where "init()" method will be executed at the time of filter initialization.

→ Where "doFilter()" method will include the actual filter logic, which we want to execute by the Container.

→ To forward request from present filter to next filter or to the service we have to use the following method from javax.servlet.FilterChain.

```
public void doFilter(ServletRequest request, ServletResponse response)
                     throws ServletException, IOException
```

Ex:- fc.doFilter(request, response);

→ Where "destroy()" method will be executed while performing Filter Deinstantiation.

WZ

## ② Configure Filter class in web.xml file:-

To Configure filter class in web.xml file we have to use the following tags in web.xml file.

<web-app>

---

<filter>

<filter-name> logical-name </filter-name>

<filter-class> Fully qualified name of the class </filter-class>

</filter>

<filter-mapping>

<filter-name> logical-name </filter-name>

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

or

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

</filter-mapping>

---

</web-app>

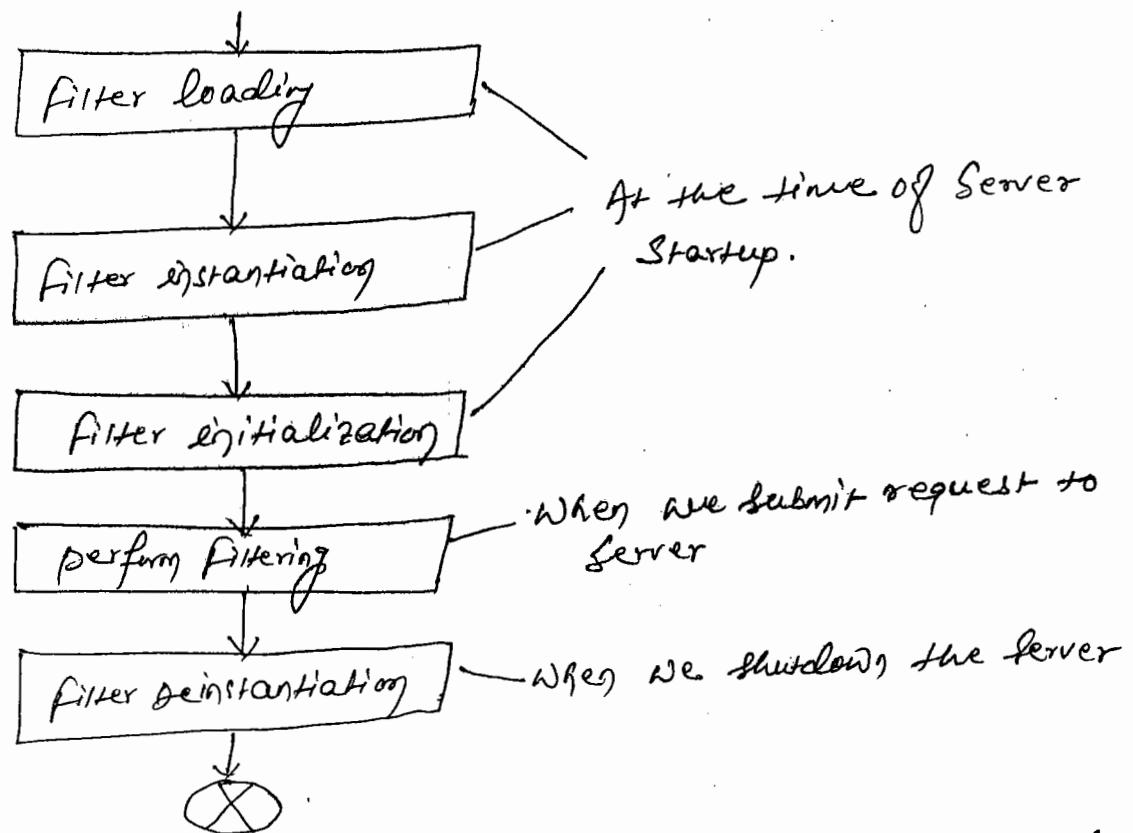
→ There are two ways to associate filters to the servlets.

① provide same servlet url pattern for filter.

② provide servlet logical name along with  
<servlet-name> tag under <filter-mapping> tag.

→ When we submit request from client to the server, Container will take the request and container will execute the filter, by using the following life cycle actions.





- In the case of filters, it is not required to provide load-on-startup Configuration, because filters are automatically loaded, instantiated and initialized at the time of Server startup.
- In we apply, we can provide a single filter to more than one servlet or more than one filter to a single servlet.
- If we provide more than one filter to a single servlet then container will execute all the filter as per <filter-mapping> tags order which we provided in web.xml file.

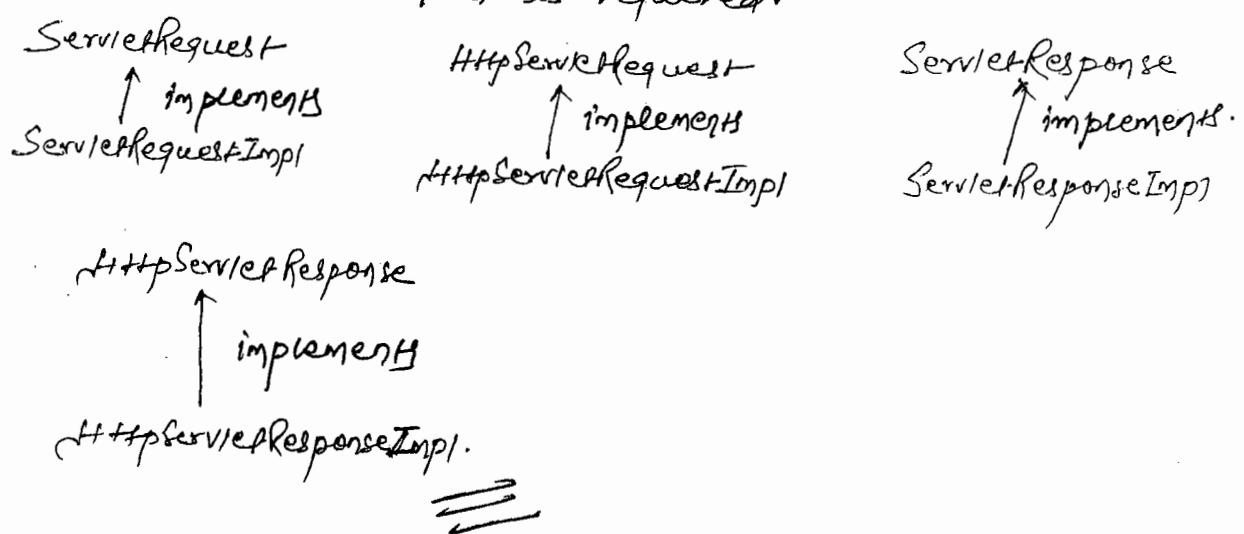
01/01/2016

## Servlet wrapper classes :-

→ As part of Servlet execution, when request is coming from Client to Server, Container will take the request, Container will identify the target-Servlet through web.xml file and Container will execute that servlet by using the service lifecycle actions like servlet loading, servlet instantiation, service initialization, Request Processing, ...

In Request Processing phase, Container must access service() method for this, Container must create request and response objects.

In web apps, request and response objects are represented by ServletRequest, HttpServletRequest, ServletResponse, HttpServletResponse interfaces provided by Sun Microsystems but their implementation classes are provided by all the server vendors. In this context, Container will create object for the Server Vendors provided implementation when it is required.



→ As per the requirement, if we want to pass our own request and our own response objects as parameters to service() method of the respective servlet then we have to prepare request and response classes.

To prepare request and response classes we have to declare user defined classes and which must implement the interfaces like ServletRequest, ServletResponse, HttpServletRequest and HttpServletResponse.

public class MyRequest implements ServletRequest {

}

public class MyResponse implements ServletResponse {

{

public class MyHttpRequest implements HttpServletRequest {

{

→ To prepare our own request and response objects if we use the above approach then we must implement all the methods of the respective interfaces in the implementation classes irrespective of the actual requirement this approach will increase unnecessary method implementing in web applications.

TC

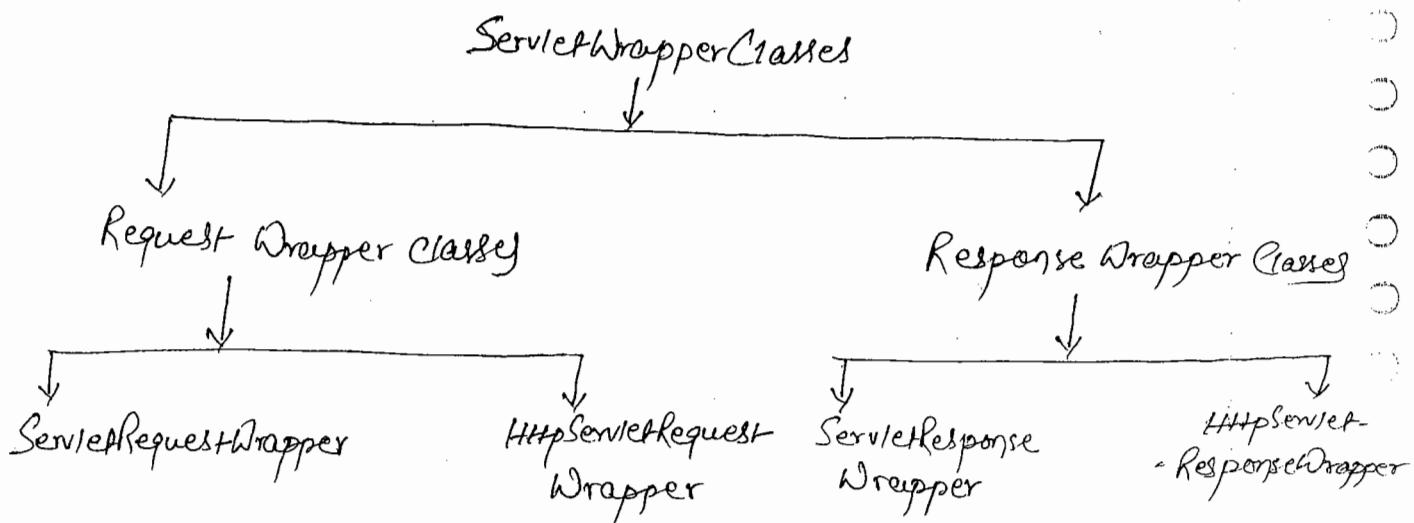
→ To overcome the above problem and to simplify the

Customization of request and response objects, Service API has provided a set of predefined classes called as "Servlet Wrapper Classes".

=

Servlet API has provided two types of wrapper classes.

- ① Request Wrapper classes
- ② Response Wrapper classes.



→ All the above wrapper classes are provided as direct implementation classes to the respective interfaces like ServletRequest, ServletResponse, HttpServletRequest and HttpServletResponse and they have provided empty implementations for all the methods of the respective interfaces.

All the servlet wrapper classes are provided by SUN Microsystems on the basis of "Adapter class" design pattern.

If we want to prepare request and response classes with Servlet Wrapper classes then we have to declare user defined classes which must be extended from the respective Servlet wrapper classes like ServletRequestWrapper,

- `HttpServletRequestWrapper`, `ServletResponseWrapper`, <sup>and</sup> `HttpServletResponseWrapper` and where we have to override only the required methods.

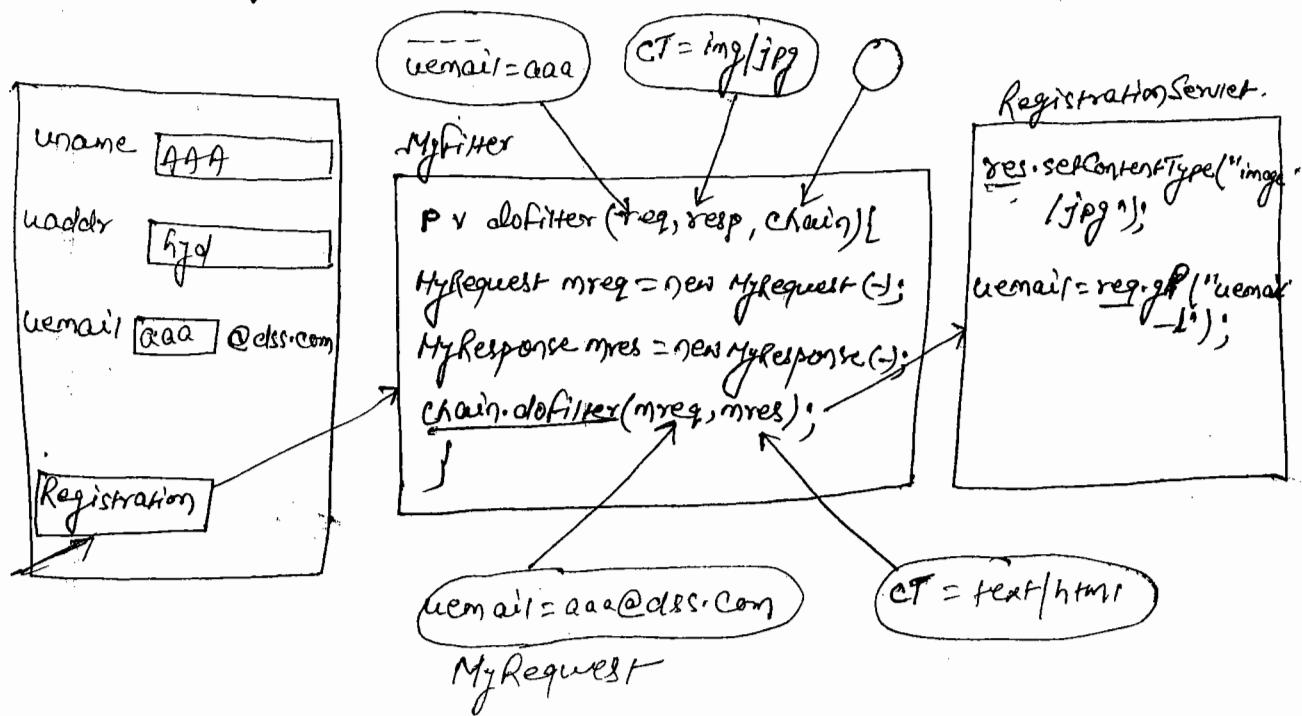
```
public class MyRequest extends ServiceRequestWrapper {  
    ---  
}
```

```
public class MyResponse extends ServiceResponseWrapper {  
    ---  
}
```

```
public class MyRequest extends HttpServletRequestWrapper {  
    ---  
}
```

```
public class MyResponse extends HttpServletResponseWrapper {  
    ---  
}
```

→ In web applications, to prepare our own request and response objects and to pass our own request and response objects to servlet they we have to use a filter.



## Listeners:-

→ In GUI applications, when we click on a button, a check box, a radio button, an item in the select box, ... automatically the respective GUI components are able to raise some events, which are not handled by the respective GUI components, to handle these events we have to use listeners.

In Event handling, the main role of the Listener is to listen events from GUI Components, handle them, perform the required actions and generating some results of the same GUI Components.

Similarly, in Web applications, Container is able to raise events at each and every lifecycle action of the objects like request, context and session, here lifecycle actions are creating objects adding an attribute, replace an attribute, removing an attribute and destroying an object.

In the above context, to handle the Container generated events, Servlet API has provided "listeners" in the form of predefined interfaces.

→ Therefore, the main purpose of the Servlet listeners is to read the lifecycle actions of the objects like request, context and session.

→ There are three types of listeners provided by Servlet API.

① Request listeners

② Context listeners

③ Session listeners.

3

## ① Request Listeners:-

→ The main purpose of the request listeners is to read the lifecycle actions of the request object like creating request object, destroying request object, adding an attribute, replacing an attribute and removing an attribute.

There are two types of request listeners.

### ① ServletRequestListeners

### ② ServletRequestAttributeListeners -

### ① ServletRequestListeners :-

→ The main purpose of this listener is to read the lifecycle actions of the request object like creation and destruction. This Listener includes the following methods to execute at the time of creating request object and at the time of destroying request object.

public void requestInitialized(ServletRequestEvent e)

public void requestDestroyed(ServletRequestEvent e)

### ② ServletRequestAttributeListener:-

→ This Listener is able to read the lifecycle actions of the request object like adding an attribute, replacing an attribute and removing an attribute.

→ To read these lifecycle actions, ~~we have to~~ this Listener has provided the following methods.

public void attributeAdded(ServletRequestAttributeEvent e)

public void attributeReplaced(ServletRequestAttributeEvent e)

public void attributeRemoved(ServletRequestAttributeEvent e)

## ② Context Listeners :-

→ The main purpose of Context Listeners is to read the lifecycle actions of the ServletContext object like Creation, destruction, adding an attribute, replacing an attribute and removing an attribute.

There are two types of Context Listeners.

### ① ServletContextListener

### ② ServletContextAttributeListener

#### ① ServletContextListener:-

→ This Listener is able to read the lifecycle actions of ServletContext Object like Creation and Destruction.

→ This Listener has provided the following methods to execute the time of reading the above lifecycle actions.

public void contextInitialized(ServletContextEvent e)  
public void contextDestroyed()

#### ② ServletContextAttributeListener:-

→ This Listener is able to read the lifecycle actions of ServletContext Object like adding an attribute, replacing an attribute and removing an attribute.

To read the above lifecycle actions, this listeners has provided the following methods.

public void attributeAdded(ServletContextAttributeEvent e)  
public void attributeReplaced()  
public void attributeRemoved()

### ③ Session Listener's:-

The main purpose of Session Listener's is to read the life cycle actions of the session object like creation, destroying, adding an attribute, replacing an attribute and removing an attribute.

There are four types of Session Listener's.

- ① HttpSessionListener
- ② HttpSessionAttributeListener
- ③ HttpSessionBindingListener.
- ④ HttpSessionActivationListener

#### ① HttpSessionListener:-

→ The main purpose of this Listener is to read the lifecycle actions of session object like creation and destruction with the following methods.

```
public void SessionCreated(HttpSessionEvent e)  
SessionDestroyed(
```

#### ② HttpSessionAttributeListener:-

→ The main purpose of this listener is to read lifecycle actions of the session object like adding an attribute, replacing an attribute and removing an attribute with the following methods.

```
public void attributeAdded (HttpSessionBindEvent e)  
public void attributeReplaced ( )  
public void attributeRemoved ( )
```

### ③ HttpSessionBindingListener:-

→ This Listener is able to read the lifecycle actions of the HttpSession object like Adding HttpSessionBindingListener implementation class object to HttpSession object and removing HttpSessionBindingListener implementation class object from HttpSession object with the following methods.

public void valueBound(HttpSessionBindingEvent e)

public void valueUnbound(HttpSessionBindingEvent e)

### ④ HttpSessionActivationListener:-

→ The main purpose of this Listener is to read the lifecycle actions of the HttpSession object like participating HttpSession object in serialization at local machine and participating HttpSession object in deserialization at remote machine in distributed applications with the following methods.

public void sessionWillPassive(HttpSessionEvent e)

public void sessionDidActive(HttpSessionEvent e)

→ If we want to use listeners in web applications then we have to use the following steps.

#### ① Create Listener class:-

→ To create Listener class, we have to declare an user defined class, it must implement the required Listener.

public class MyListener implements XXXListener {

— Implementation for Listener interface methods —

}

=

## ② Configure Listener Class in web.xml file:-

```
<web-app>
  <listeners>
    <listener>
      <listener-class>MyListener</listener-class>
    </listener>
  </listeners>
</web-app>
```

Note:- The above web.xml file configuration is required for all the listeners accept HttpSessionBindingListener and HttpSessionActivationListener.  
Ex:- HttpSessionListener (in material).

## Annotations in Servlet [Servlet 3.0 Version]

- In web applications upto servlet 2.5v web.xml file is mandatory to execute servlets, filters, listeners,...
- In servlet 3.0v to execute the server side components like Servlets, filters, listeners, ... we are able to use annotations as an alternatives to web.xml file.
- To provide metadata about the server side components like servlets, filters, listeners, ... without using web.xml file servlet 3.0v has provided the following annotations in javax.servlet.annotation package.

- ① @WebServlet
- ② @WebInitParam
- ③ @WebFilter
- ④ @WebListener

## ① @WebServlet:-

This annotation will provide the metadata about a particular Servlet which required by the container in order to execute Servlets.

### Syntax:-

`@WebServlet`

`name = " - " ,`

`urlPatterns = "[ " , " " , .. ] ,`

`loadOnStartup = - ,`

`initParams = { @WebInitParam ( name = " -- " , value = " -- " ) ,`

`@WebInitParam ( name = " -- " , value = " -- " ) , .. } }`

Where "name" member will take Servlet logical name.

Where "urlPatterns" member is able to take an integer value to perform servlet loading, instantiation and initialization at the time of server startup.

Where "initParams" members will take array of `@WebInitParam` annotations to provide initialization parameters.

## ② @

→ This annotation will represent a single initialization parameter.

### Syntax:-

`@WebInitParam ( name = " -- " , value = " -- " )`

→ Where "name" and "value" members are able to provide initialization parameter name and value.

### ③ @WebFilter:-

→ This annotation will provide metadata about a filter which is required by the container to execute.

Syntax:-

@WebFilter

```
serviceName="--",
name="--",
urlPatterns={"--", "- ", ...})
```

→ Where "serviceName" member will take service logical name to which service we want to provide this filter.

Where "name" member will take logical name of the filter.  
Where "urlPatterns" members will take array of url patterns to the filter.

### ④ @WebListener:-

→ It is a marker annotation, it will provide listener configuration for a particular Listener.

Syntax:-

@WebListener

```
public class MyListener implements XXXListener {
```

```
}
```

## File Uploading:-

File uploading is a frequently used operations in web applications in order to upload some resources from Client to Server.

- To implement file uploading in web applications we have to use some third party libraries like "Apache Commons File Uploading" library. Apache has provided file uploaded library in the form of the following two jar files, these two jar files must be provided in web application lib folder.

Commons - io - 1.1.1.jar

Commons - fileupload - 1.1.1.jar

- To implement file uploading in web applications then we have to use the following steps

- ① Create an User form with file components
- ② Create a service to perform uploading.

- Declare an user form with `<form>` tag.
- In user form "method" attribute value must be "post".
- In user form provide "multipart/form-data" to "enctype" attribute.
- In user form, declare file component by using `<input type = "file">` tag.
- Create a button.

Ex:-

```
uploadform.htm
<html>
  <body>
```

```
    <form action = "/upload" method = "post" enctype = "multipart/form-data">
```

Servlet - 01/01/16

```
select file<input type="file" name="file"/>
<input type="submit" value="upload"/>
</form>
</body>
</html>
```

result.html

```
<html>
<body>
<h1>File uploaded successfully!</h1>
<h2><a href=". /uploadform.html">One more file</a></h2>
</body>
</html>
```

## ② Create a Servlet to upload resources :-

### ① Create DiskfileItemfactory class object.

```
DiskfileItemfactory dfi = new DiskfileItemfactory();
```

### ② Create ServletfileUpload class object:

```
ServletfileUpload sfu = new ServletfileUpload(dfi);
```

### ③ Get all the uploaded file in the form of FileItem objects.

```
List<FileItem> list = sfu.parseRequest(request);
```

### ④ Get data from FileItem object and send to a particular target file by using the following method.

```
public void write(File target_file)
```

≡

Cat-

### FileUploadServlet.java

```
import java.io.*;  
import java.util.List;  
import javax.servlet.*;  
import javax.servlet.http.*;  
import org.apache.commons.fileupload.FileItem;  
import org.apache.commons.fileupload.disk.DiskFileItemFactory;  
import org.apache.commons.fileupload.servlet.ServletFileUpload;
```

```
public class FileUploadServlet extends HttpServlet {  
    protected void doPost(HttpServletRequest request, HttpServletResponse  
        response) throws ServletException, IOException,
```

```
{
```

```
    DiskFileItemFactory dfi = new DiskFileItemFactory();  
    ServletFileUpload sfu = new ServletFileUpload(dfi);  
    List<FileItem> multiparts = sfu.parseRequest(request);  
    for (FileItem item : multiparts) {  
        File f = new File(item.getName());  
        String name = f.getName();  
        item.write(new File("D:/uploads/" + name));  
    }  
    RequestDispatcher rd = request.getRequestDispatcher("/result.html");  
    rd.forward(request, response);  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

In case of Dynamic Form Generation, we will define user form a servlet. If we require Dynamic form then we have to access required respective servlet.

&lt;/se

&lt;/w

Hea

Headersapp:

web.xml

pack

-----

imp

```
<?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>headersapp</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <description></description>
    <display-name>HeadersServlet</display-name>
    <servlet-name>HeadersServlet</servlet-name>
    <servlet-class>com.durgasoft.HeadersServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HeadersServlet</servlet-name>
    <url-pattern>/headers</url-pattern>
```

imp

imp

imp

imp

impo

impo

impo

publi

throv

align



```
out.println("<tr><td>" + header_Name + "</td><td>" + header_Value + "</td></tr>");
```

```
}
```

```
out.println("</table></center></body></html>");
```

```
}
```

```
}
```

Parametersapp:

---

registrationform.java

---

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="ISO-8859-1">
```

```
<title>Insert title here</title>
```

```
</head>
```

```
<body>
```

```
<font color="red">
```

```
<h2>Durga Software Solutions</h2>
```

```
<h3>Student Registration Form</h3>
```

```
</font>
```

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

```
<table>
```

```
<tr>
```

```
<td>Student Id</td>
```

```
<td><input type="text" name="sid"/></td>
```

```
</tr>
```

```
<tr>
```



Loginapp:

layout.html

```
<!DOCTYPE html>
<frameset rows="20%,65%,15%">
    <frame src="header.html"/>
    <frameset cols="20%,80%">
        <frame src="menu.html"/>
        <frame src="welcome.html" name="body"/>
    </frameset>
    <frame src="footer.html"/>
</frameset>
```

Header.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body bgcolor="maroon">
<center>
<font color="white" size="7">
<b>
DURGA SOFTWARE SOLUTIONS
</b>
</font>
```



```
        status="success";  
    }  
}  
} catch (Exception e) {  
    status="failure";  
    e.printStackTrace();  
}  
return status;  
}  
  
}
```

Dynamicformapp:

updateform.html

---

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body>  
<font color="red">  
<h2>Durga Software Solutions</h2>  
<h3>Student Update Form</h3>  
</font>  
<form method="GET" action=".edit">  
<table>  
<tr>
```



```
<h2>Durga Consultancy Services</h2>
<h3>User Registration Form</h3>
</font>
<form method="POST" action=".reg">
<table>
<tr>
    <td>User Name</td>
    <td><input type="text" name="uname"/></td>
</tr>
<tr>
    <td>User Age</td>
    <td><input type="text" name="uage"/></td>
</tr>
<tr>
    <td>User Email</td>
    <td><input type="text" name="uemail"/></td>
</tr>
<tr>
    <td>User Mobile</td>
    <td><input type="text" name="umobile"/></td>
</tr>
<tr>
    <td><input type="submit" value="Registration"/></td>
</tr>
</table>
```



```

<tag>
<name>custom tag name</name>
<tag-class>fully qualified name of TagHandler class</tag-class>
<body-content>jsp or empty</body-content>
<short-name>custom tag short name</short-name>
<description>description about custom tags</description>
</tag>
-----
</taglib>

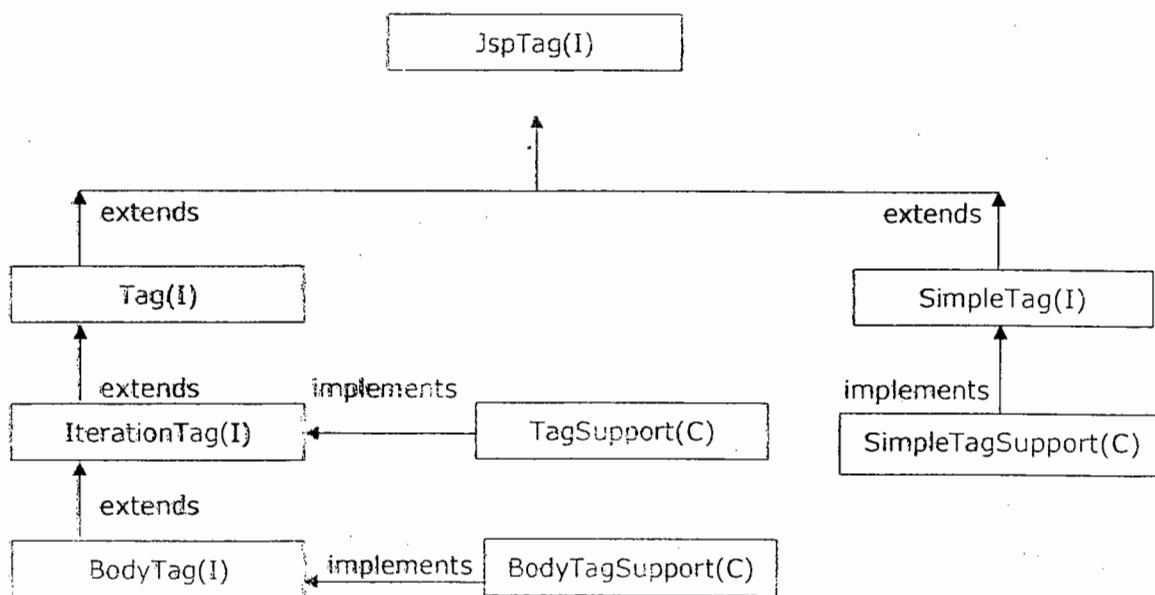
```

### 3. TagHandler class:

In custom tags preparation, the main purpose of TagHandler class is to define the basic functionality for the custom tags.

To design TagHandler classes in custom tags preparation Jsp API has provided some predefined library in the form of javax.servlet.jsp.tagext package (tagext→tag extension).

javax.servlet.jsp.tagext package has provided the following library to design TagHandler classes.



The above Tag library was divided into 2 types.

1. Classic Tag Library
2. Simple Tag Library

As per the tag library provided by Jsp technology there are 2 types of custom tags.

1. Classic tags



In case of Dynamic Form Generation, we will define user form a servlet. If we require Dynamic form then we have to access required respective servlet.

Headersapp:

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>headersapp</display-name>

  <welcome-file-list>

    <welcome-file>index.html</welcome-file>

    <welcome-file>index.htm</welcome-file>

    <welcome-file>index.jsp</welcome-file>

    <welcome-file>default.html</welcome-file>

    <welcome-file>default.htm</welcome-file>

    <welcome-file>default.jsp</welcome-file>

  </welcome-file-list>

  <servlet>

    <description></description>

    <display-name>HeadersServlet</display-name>

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

    <servlet-class>com.durgasoft.HeadersServlet</servlet-class>

  </servlet>

  <servlet-mapping>

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

    <url-pattern>/headers</url-pattern>
```

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

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

```
HeadersServlet.java
```

```
-----  
package com.durgasoft;
```

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import java.util.Enumeration;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

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

```
public class HeadersServlet extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
throws ServletException, IOException {
```

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

```
        PrintWriter out=response.getWriter();
```

```
        Enumeration<String> e=request.getHeaderNames();
```

```
        out.println("<html>");
```

```
        out.println("<body><center><br><br>");
```

```
        out.println("<table border='1' bgcolor='lightblue'>");
```

```
        out.println("<tr><td align='center'><h3>Header Names</h3></td><td  
align='center'><h3>Header Values</h3></td></tr>");
```

```
        while(e.hasMoreElements()){
```

```
            String header_Name=(String)e.nextElement();
```

```
            String header_Value=request.getHeader(header_Name);
```

```
out.println("<tr><td>" + header_Name + "</td><td>" + header_Value + "</td></tr>");  
}  
out.println("</table></center></body></html>");  
}  
}
```

Parametersapp:

registrationform.java

---

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body>  
<font color="red">  
<h2>Durga Software Solutions</h2>  
<h3>Student Registration Form</h3>  
</font>  
<form method="POST" action=".reg">  
<table>  
<tr>  
<td>Student Id</td>  
<td><input type="text" name="sid"/></td>  
</tr>  
<tr>
```

```
<td>Student Name</td>
<td><input type="text" name="sname"/></td>
</tr>
<tr>
<td>Student Qualification</td>
<td>
<input type="checkbox" value="BSC" name="squal"/>BSC<br>
<input type="checkbox" value="MCA" name="squal"/>MCA<br>
<input type="checkbox" value="PHD" name="squal"/>PHD<br>
</td>
</tr>
<tr>
<td>Student Gender</td>
<td>
<input type="radio" value="Male" name="sgender"/>Male<br>
<input type="radio" value="Female" name="sgender"/>Female<br>
</td>
</tr>
<tr>
<td>Student Technologies</td>
<td>
<select size="5" name="stech" multiple="multiple">
<option value="C">C</option>
<option value="C++">C++</option>
<option value="Java">JAVA</option>
<option value=".Net">.Net</option>
<option value="Oracle">Oracle</option>

```

```
<option value="Testing Tools">Testing Tools</option>
</select>
</td>
</tr>
<tr>
<td>Branch</td>
<td>
<select name="branch">
<option value="Ameerpet">Ameerpet</option>
<option value="S R Nagar">S R Nagar</option>
<option value="Madapur">Madapur</option>
<option value="KPHB">KPHB</option>
</select>
</td>
</tr>
<tr>
<td>Student Address</td>
<td><textarea rows="10" cols="50" name="saddr"></textarea></td>
</tr>
<tr>
<td><input type="submit" value="Registration"/></td>
</tr>
</table>
</form>
</body>
</html>
```

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>parametersapp</display-name>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <servlet>
    <description></description>
    <display-name>RegistrationServlet</display-name>
    <servlet-name>RegistrationServlet</servlet-name>
    <servlet-class>com.durgasoft.RegistrationServlet</servlet-class>
  </servlet>

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

</web-app>
```

## RegistrationServlet.java

```
-----  
package com.durgasoft;  
  
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 RegistrationServlet extends HttpServlet {  
  
    private static final long serialVersionUID = 1L;  
  
    protected void doPost(HttpServletRequest request, HttpServletResponse response)  
throws ServletException, IOException {  
  
        response.setContentType("text/html");  
  
        PrintWriter out=response.getWriter();  
  
        String sid=request.getParameter("sid");  
  
        String sname=request.getParameter("sname");  
  
        String[] squal=request.getParameterValues("squal");  
  
        String sgender=request.getParameter("sgender");  
  
        String[] stech=request.getParameterValues("stech");  
  
        String branch=request.getParameter("branch");  
  
        String saddr=request.getParameter("saddr");  
  
        String qual="";  
  
        for(int i=0;i<squal.length;i++){  
  
            qual=qual+squal[i]+"<br>";  
        }  
    }  
}
```

```
String tech="";
for(int j=0;j<stech.length;j++){
    tech=tech+stech[j]+<br>;
}
out.println("<html>");
out.println("<body>");
out.println("<font color='red'>");
out.println("<h2>Durga Software Solutions</h2>");
out.println("<h3>Student Registration Details</h3>");
out.println("</font>");
out.println("<table border='1'>");
out.println("<tr><td>Student Id</td><td>" + sid + "</td></tr>");
out.println("<tr><td>Student Name</td><td>" + sname + "</td></tr>");
out.println("<tr><td>Student Qualification</td><td>" + qual + "</td></tr>");
out.println("<tr><td>Student Gender</td><td>" + sgender + "</td></tr>");
out.println("<tr><td>Student Technologies</td><td>" + tech + "</td></tr>");
out.println("<tr><td>Branch</td><td>" + branch + "</td></tr>");
out.println("<tr><td>Student Address</td><td>" + saddr + "</td></tr>");
out.println("</table></body></html>");
}

}
```

Loginapp:

layout.html

```
<!DOCTYPE html>
<frameset rows="20%,65%,15%">
    <frame src="header.html"/>
    <frameset cols="20%,80%">
        <frame src="menu.html"/>
        <frame src="welcome.html" name="body"/>
    </frameset>
    <frame src="footer.html"/>
</frameset>
```

Header.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body bgcolor="maroon">
<center>
<font color="white" size="7">
<b>
DURGA SOFTWARE SOLUTIONS
</b>
</font>
```

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

menu.html

---

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body bgcolor="lightyellow">  
<br><br>  
<h3>  
<a href="loginform.html" target="body">Login</a>  
<br><br>  
<a href="registrationform.html" target="body">Registration</a>  
</h3>  
</body>  
</html>
```

welcome.html

---

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

```
<title>Insert title here</title>
</head>
<body bgcolor="lightblue">
<center>
<font color="red" size="6">
<b><br><br><br>
<marquee>
Welcome To Durga Software Solutions
</marquee>
</b>
</font>
</center>

</body>
</html>
```

footer.html

---

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body bgcolor="blue">
<center>
<font color="white" size="5">
```

```
<b>  
copyright reserved @ Durga Software Solutions, S R Nagar  
</b>  
</font>  
</center>  
  
</body>  
</html>
```

loginform.html

```
-----  
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body bgcolor="lightblue">  
<br><br>  
<form method="POST" action=".//login">  
<center>  
<table>  
<tr>  
    <td>User Name</td>  
    <td><input type="text" name="uname"/></td>  
</tr>  
<tr>
```

```
<td> Password</td>
<td><input type="password" name="upwd"/></td>
</tr>
<tr>
<td><input type="submit" value="Login"/></td>
</tr>
</table>
</center>
</form>
</body>
</html>
```

registrationform.html

---

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body bgcolor="lightblue">
<br><br>
<form method="POST" action=".//reg">
<center>
<table>
<tr>
<td>User Name</td>
<td><input type="text" name="uname"/></td>
```

```
</tr>
<tr>
    <td>Password</td>
    <td><input type="password" name="upwd"/></td>
</tr>
<tr>
    <td>User Email</td>
    <td><input type="text" name="uemail"/></td>
</tr>
<tr>
    <td>User Mobile Num</td>
    <td><input type="text" name="umobile"/></td>
</tr>

<tr>
    <td><input type="submit" value="Registration"/></td>
</tr>
</table>
</center>
</form>
</body>
</html>
```

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>loginapp</display-name>

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <servlet>
    <description></description>
    <display-name>LoginServlet</display-name>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>com.durgasoft.LoginServlet</servlet-class>
  </servlet>

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

  <servlet>
    <description></description>
```

```
<display-name>RegistrationServlet</display-name>
<servlet-name>RegistrationServlet</servlet-name>
<servlet-class>com.durgasoft.RegistrationServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>RegistrationServlet</servlet-name>
<url-pattern>/reg</url-pattern>
</servlet-mapping>
</web-app>
```

LoginServlet.html

---

```
package com.durgasoft;
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 LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String uname=request.getParameter("uname");
```

```
String upwd=request.getParameter("upwd");
UserService us=new UserService();
String status=us.checkLogin(uname,upwd);
out.println("<html>");
out.println("<body bgcolor='lightblue'>");
out.println("<center><br><br>");
out.println("<font color='red' size='7'>");
if(status.equals("success")){
    out.println("Login Success");
}
if(status.equals("failure")){
    out.println("Login Failure");
}
out.println("</font></center></body></html>");

}
```

### RegistrationServlet.java

---

```
package com.durgasoft;
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 RegistrationServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String uname=request.getParameter("uname");
        String upwd=request.getParameter("upwd");
        String uemail=request.getParameter("uemail");
        String umobile=request.getParameter("umobile");
        UserService us=new UserService();
        String status=us.registration(uname,upwd,uemail,umobile);
        out.println("<html>");
        out.println("<body bgcolor='lightblue'>");
        out.println("<center><br><br>");
        out.println("<font color='red' size='7'>");
        if(status.equals("success")){
            out.println("Registration Success");
        }
        if(status.equals("failure")){
            out.println("Registration Failure");
        }
        if(status.equals("existed")){
            out.println("User Existed Already");
        }
    }
}
```

```
    }

    out.println("</font></center></body></html>");

}

}
```

**UserService.java**

```
-----
package com.durgasoft;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class UserService {

    Connection con;
    Statement st;
    ResultSet rs;
    String status="";

    public UserService() {
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"durga");
            st=con.createStatement();
        } catch (Exception e) {
```

```
e.printStackTrace();  
}  
}  
  
public String checkLogin(String uname, String upwd){  
    try {  
        rs=st.executeQuery("select * from reg_Users where  
        uname='"+uname+"' and upwd='"+upwd+"'");  
        boolean b=rs.next();  
        if(b==true){  
            status="success";  
        }else{  
            status="failure";  
        }  
    } catch (Exception e) {  
    }  
    return status;  
}  
  
public String registration(String uname, String upwd, String uemail, String umobile){  
    try {  
        rs=st.executeQuery("select * from reg_Users where  
        uname='"+uname+"'");  
        boolean b=rs.next();  
        if(b==true){  
            status="existed";  
        }else{  
            st.executeUpdate("insert into reg_Users  
            values('"+uname+"','"+upwd+"','"+uemail+"','"+umobile+"')");  
        }  
    } catch (Exception e) {  
    }  
}
```

```
        status="success";  
    }  
  
} catch (Exception e) {  
    status="failure";  
    e.printStackTrace();  
}  
  
return status;  
}  
  
}
```

---

**Dynamicformapp:**

updateform.html

---

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body>  
<font color="red">  
<h2>Durga Software Solutions</h2>  
<h3>Student Update Form</h3>  
</font>  
<form method="GET" action=".edit">  
<table>  
<tr>
```

```
<td>Student Id</td>
<td><input type="text" name="sid"/></td>
</tr>
<tr>
    <td><input type="submit" value="GetEditForm"/></td>
</tr>
</table>
</form>
</body>
</html>
```

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>dynamicformapp</display-name>  
    <welcome-file-list>  
        <welcome-file>updateform.html</welcome-file>  
    </welcome-file-list>  
    <servlet>  
        <description></description>  
        <display-name>EditFormServlet</display-name>  
        <servlet-name>EditFormServlet</servlet-name>  
        <servlet-class>com.durgasoft.EditFormServlet</servlet-class>  
    </servlet>
```

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

<servlet>
<description></description>
<display-name>UpdateServlet</display-name>
<servlet-name>UpdateServlet</servlet-name>
<servlet-class>com.durgasoft.UpdateServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>UpdateServlet</servlet-name>
<url-pattern>/update</url-pattern>
</servlet-mapping>
</web-app>
```

**EditFormServlet.java**

```
-----
package com.durgasoft;
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 EditFormServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        String sid=request.getParameter("sid"); → here, we need to get object from request object
        StudentService ss=new StudentService();
        StudentTo sto=ss.getStudent(sid); → java bean class
        if(sto==null){
            out.println("<html>");
            out.println("<body>");
            out.println("<br><br>"); → out.println( "<h2> Student not Existed(1)</h2>" );
            out.println("<font color='red' size='6'>"); → or
            out.println("Student Not Existed");
            out.println("</font>");
            out.println("<br><br>");
            out.println("<h3><a href='./updateform.html'>|Update Form|</a></h3>"); → out.println(" |Update Form| ");
            out.println("</body></html>");
        }else{
            out.println("<html>");
            out.println("<body>");
            out.println("<font color='red'>"); → out.println(" <font color='red'> ");
            out.println("<h2>Durga Software Solutions</h2>"); → out.println(" Durga Software Solutions ");
            out.println("<h3>Student Edit Form</h3>"); → out.println(" Student Edit Form ");
            out.println("</font>"); → out.println(" </font> ");
        }
    }
}
```

↑ POST

```

out.println("<form method='GET' action='./update'>");

out.println("<table>");

out.println("  <tr><td>Student Id</td><td>" + sid + "</td></tr>");

out.println("  <tr><td>Student Name</td><td><input type='text' name='sname' value='" + sto.getName() + "'></td></tr>");

out.println("  <tr><td>Student Address</td><td><input type='text' name='saddr' value='" + sto.getAddress() + "'></td></tr>");

out.println("  <tr><td><input type='submit' value='Update'></td></tr>");

out.println("</table></form></body></html>");

}

}

```

UpdateServlet.java

---

```

package com.durgasoft;

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 UpdateServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    String sid=request.getParameter("sid");
    String sname=request.getParameter("sname");
    String saddr=request.getParameter("saddr");
    StudentService ss=new StudentService();
    String status=ss.update(sid,sname,saddr);
    out.println("<html>");
    out.println("<body>");
    out.println("<h2>");
    out.println("<br><br>");
    if(status.equals("success")){
        out.println("Student Updated Successfully<br>");
    }else{
        out.println("Student Updation Failure<br>");
    }
    out.println("<a href='./update_form.html'>|Update Form|</a>");
    out.println("</h2></body></html>");
}
```

StudentService.java

---

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

```
import java.sql.ResultSet;
import java.sql.Statement;

public class StudentService {
    Connection con;
    Statement st;
    ResultSet rs;
    String status="";
    StudentTo sto;
    public StudentService() {
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","durga");
            st=con.createStatement();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public StudentTo getStudent(String sid){
        try {
            rs=st.executeQuery("select * from student where sid='"+sid+"'");
            boolean b=rs.next();
            if(b==true){
                sto=new StudentTo();
                sto.setSid(rs.getString(1));
                sto.setSname(rs.getString(2));
            }
        }
    }
}
```

"SID" → "STUDENT"  
"SNAME" → "STUDENT"

```
sto.setSaddr(rs.getString(3));  
} else {  
    sto=null;  
}  
}  
} catch (Exception e) {  
  
}  
return sto;  
}  
  
public String update(String sid, String sname, String saddr){  
try {  
    st.executeUpdate("update student set  
    sname='"+sname+"',saddr='"+saddr+"' where sid='"+sid+"'");  
    status="success";  
} catch (Exception e) {  
    status="failure";  
    e.printStackTrace();  
}  
return status;  
}  
}
```

StudentTo.java

---

```
package com.durgasoft;  
public class StudentTo {
```

```
private String sid;
private String sname;
private String saddr;
public String getSid() {
    return sid;
}
public void setSid(String sid) {
    this.sid = sid;
}
public String getSname() {
    return sname;
}
public void setSname(String sname) {
    this.sname = sname;
}
public String getSaddr() {
    return saddr;
}
public void setSaddr(String saddr) {
    this.saddr = saddr;
}
}
```

## Welcome Files:

In general, in all the web applications some pages like login pages, index pages, home pages and so on are the first pages.

In the above context, to access the first pages we have to specify the respective html page name or jsp page name as resource name in URL even though they are common for each and every user.

To overcome the problem, we have to declare the respective html or jsp page as welcome file.

**Welcome file** is the first page of the web application, it must be executed by the container automatically when we access the respective application without specifying resource name in URL.

To declare welcome file in web.xml file, we have to use the following xml tags.

**Ex:** <web-app>  
<welcome-file-list>  
<welcome-file>file1</welcome-file>  
<welcome-file>file1</welcome-file>  
-----  
</welcome-file-list>  
-----  
</web-app>

From the above tags representation, it is possible to provide more than one welcome file with in a single web application but w.r.t. multiple no. of modules.

If we provide more than one welcome file with in a single web application w.r.t. modules the container will search for the respective welcome file as per the order in which we configured web.xml file.

## Smooth Deployment:

In general, we will prepare web applications with Tomcat server by creating the entire web application directly structure under webapps folder.

In this case, when we start the server then automatically the prepared web application will be deployed into the server.

## 1. Browser-Servlet Communication:-

In general in web applications, we will use browser as a client at client machine, from the browser we will send a request to a servlet available at server, where the servlet will be executed and generate some response to the client browser.

In the above process, we have provided communication between client browser and servlet, so that sending a normal request from client to server and getting a normal response from server to client is an example for **Browser-Servlet Communication**.

## 2. Sending Error Messages:

As part of the web application execution, if the container identify any exception or error then container will send the respective error message to client in its standalone template.

As part of our application, if we want to send our own messages to the client in the container defined template we have to use the following method from response.

```
public void sendError(int statuscode, String description)
```

where statuscode may be 5xx.

SendErrorApp:

Registrationform.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<font color='red'>
```

```
<h2>Durga Consultency Services</h2>
<h3>User Registration Form</h3>
</font>
<form method="POST" action=".reg">
<table>
<tr>
    <td>User Name</td>
    <td><input type="text" name="uname"/></td>
</tr>
<tr>
    <td>User Age</td>
    <td><input type="text" name="uage"/></td>
</tr>
<tr>
    <td>User Email</td>
    <td><input type="text" name="uemail"/></td>
</tr>
<tr>
    <td>User Mobile</td>
    <td><input type="text" name="umobile"/></td>
</tr>
<tr>
    <td><input type="submit" value="Registration"/></td>
</tr>
</table>
```

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

---

**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>senderrorapp</display-name>  
    <welcome-file-list>  
        <welcome-file>registrationform.html</welcome-file>  
    </welcome-file-list>  
    <servlet>  
        <description></description>  
        <display-name>RegistrationServlet</display-name>  
        <servlet-name>RegistrationServlet</servlet-name>  
        <servlet-class>com.durgasoft.RegistrationServlet</servlet-class>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>RegistrationServlet</servlet-name>  
        <url-pattern>/reg</url-pattern>  
    </servlet-mapping>  
</web-app>
```

```
<h2>Durga Consultancy Services</h2>
<h3>User Registration Form</h3>
</font>

<form method="POST" action=".//reg">
<table>
<tr>
    <td>User Name</td>
    <td><input type="text" name="uname"/></td>
</tr>
<tr>
    <td>User Age</td>
    <td><input type="text" name="uage"/></td>
</tr>
<tr>
    <td>User Email</td>
    <td><input type="text" name="uemaii"/></td>
</tr>
<tr>
    <td>User Mobile</td>
    <td><input type="text" name="umobile"/></td>
</tr>
<tr>
    <td><input type="submit" value="Registration"/></td>
</tr>

</table>
```

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

---

**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>senderrorapp</display-name>

  <welcome-file-list>
    <welcome-file>registrationform.html</welcome-file>
  </welcome-file-list>

  <servlet>
    <description></description>
    <display-name>RegistrationServlet</display-name>
    <servlet-name>RegistrationServlet</servlet-name>
    <servlet-class>com.durgasoft.RegistrationServlet</servlet-class>
  </servlet>

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

</web-app>
```

RegistrationServlet.java

```
package com.durgasoft;

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 RegistrationServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        try {
            response.setContentType("text/html");
            PrintWriter out=response.getWriter();
            String uname=request.getParameter("uname");
            int uage=Integer.parseInt(request.getParameter("uage"));
            String uemail=request.getParameter("uemail");
            String umobile=request.getParameter("umobile");

            if(uage<18 || uage>30){
                response.sendError(504, "User Age Is Not Sufficient for this
Recruitment");
            }else{
                out.println("<html>");
                out.println("<body>");
                out.println("<font color='red'>");
                out.println("<h2>Durga Consultancy Services</h2>");
            }
        }
    }
}
```

```
out.println("<h3>User Registration Details</h3>");

out.println("</font>");

out.println("<table border='1'>");

out.println("<tr><td>User
Name</td><td>" + uname + "</td></tr>");

out.println("<tr><td>User
Age</td><td>" + uage + "</td></tr>");

out.println("<tr><td>User
Email</td><td>" + uemail + "</td></tr>");

out.println("<tr><td>User
Mobile</td><td>" + umobile + "</td></tr>");

out.println("</table></body></html>");

}

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

}

}
```



JSP (Java Server Pages):-

There are two types of responses we are able to get from web applications.

- (1) Static Response
- (2) Dynamic Response.

(1) Static Response:-

If we generate response from Server machine without performing any action or without executing any resource at Server machine then that response is called as static response.

Ex:- Accessing an HTML file available at server from client.

(2) Dynamic Response:-

If generate any response from server by executing a particular resource or by performing an action at server machine then that response is called as dynamic response.

Ex:- Accessing a servlet or JSP page available at server from Client.

To perform an action at server we have to prepare an app called as web app. To create web application at server we have to use a set of technologies called as Web technologies.

Ex:- CGI, Servlets, JSPs, ...

→ CGI is a server side technology, designed on the basis of C programming language, it is a process based technology, CGI Container will create a separate process for each and every request coming from Client and it will reduce server side application performance, because process is heavy weight component.

→ Servlet is a server side technology, designed on the basis of J2EE/JAVA SE, it is a thread based technology, Servlet container will create a separate thread for each and every request coming from Client and it will improve Server side application performance, because threads are light weight components when compared with processes.

→ If we want to design web apps, by using Servlet the developer must have very good awareness on JAVA. If we want to design web applications by using JSP tech then it is not all required awareness on JAVA.

The main utilization of JSP tech is to reduce java code in web applications as much as possible.

In web applications, we will use servlets at the time of taking request from a client and at the time of ~~taking requests~~ processing requests. In web apps we will use JSP tech at the time of generating ~~dynamic~~ response from server to client with very good look and feel.

The main utilization of JSP tech in web apps is to prepare presentation part.

In MVC based web apps, we have to use a servlet as controller and a set of JSP pages as view part.

Ex: ① Struts framework is MVC based framework, it will use ActionServlet as controller and a set of JSP pages as view part.

Ex: ② JSF is MVC based framework, it will use FaceServlet as controller and a set of JSP pages as view part.

- JSP tech is designed on the basis of J2SE API and Servlets API.
- In web apps, JSP pages are executing on the top of servlets only, because every JSP page is translated to a servlet by the container in order to execute JSP page.

====

- In web applications, we are able to provide JSP pages at any location of the web application directory structure, but always it is suggestable to provide JSP pages under app's folder.
- If we provide JSP page under application folder ~~the~~ [public Area/Client Area] then we are able to access that JSP by providing its name directly at client address bar along with URL.
- If we provide JSP page under WEB-INF or under classes locations [Private Area/Server Area] then to access JSP page we must define an URL pattern for JSP page in web.xml file and we have to use the defined URL pattern at client address bar along with URL.

To provide JSP page configuration in web.xml file we have to use the following XML tags in web.xml file.

```
<web-app>
  < servlet >
    < servlet-name > logical Name < />
    < JSP - file > Context relative path to jsp file<
  </servlet>
  < servlet-mapping >
    < servlet-name > logical Name < />
    < url-pattern > pattern1 < />
    < url-pattern > pattern2 < />
```

---

---

/servlet-mapping

{/web-app}

Z

Ex:-

firstjspapp

| → welcome.jsp

| → WEB-INF

  | → classes

welcome.jsp

(html)

(body)

}

  <b>Welcome To Durga Software Solutions</b>

(hr)

(html)

(body)

(html)

firstjspapp1

|

| → WEB-INF

  | → Web.XML

  | → classes

  | → welcome.jsp

Welcome.jsp

same as previous application

## web.xml

```
<web-app>
  <service>
    <s-n> WelcomeTspage</s-n>
    <J-f>/WEB-INF/classes/welcome.jsp</
  <service>
  <service-mapping>
    <s-n> WelcomeTspage</s-n>
    <url-pattern>/abc</url-pattern>
  </service-mapping>
</web-app>
```

Σ

30/12/15  
31/12/15

## JSP Lifecycle :-

When we start server Container will perform the following actions.

- ① Container will recognize all the web applications which are available in webapps[Tomcat] folder.
  - ② Container will deploy all these web applications into server space.
  - ③ Container will create a separate ServletContext Object for each and every deployed application.
- While recognizing web applications, Container will search for web.xml file, if it is available then Container will perform web.xml file loading, parsing and reading the content.  
In web.xml file, if any application level data is identified then that data will be stored in the respective ServletContext objects.

Σ

→ When we submit request from Client to Server, protocol will take request and protocol will perform the following actions.

- ① Protocol will establish logical connection between client and server as per the server IP Address and port no. which we specified in url.
- ② Protocol will prepare request format with header part and body part, where header part is able to manage request headers data and body part is able to manage request parameters data.
- ③ After getting Requestformat, protocol will carry Requestformat to server.

→ When request is reached to server, Main Server will take the request, Main Server will check whether request is proper or not, if it is not proper then Main Server will send error response to client, if the request is proper then Main Server will forward request to container.

→ When request is reached to container, Container will take application name and resource name from request and Container will check whether the resource is any JSP page or URL pattern, if it is JSP page then Container will search for the respective JSP page under application folder, if it is available then Container will execute that JSP file by using the following lifecycle actions.

~~Request processing~~

Cont..

### ① JSP Loading:-

→ Container will load JSP file content to the memory.

### ② JSP parsing:-

→ Container will checks all the dynamics of the tags which we used in JSP file.

### ③ JSP file translation to Servlet:-

→ Container will translate JSP file into the equivalent Servlet with the JSP file content.

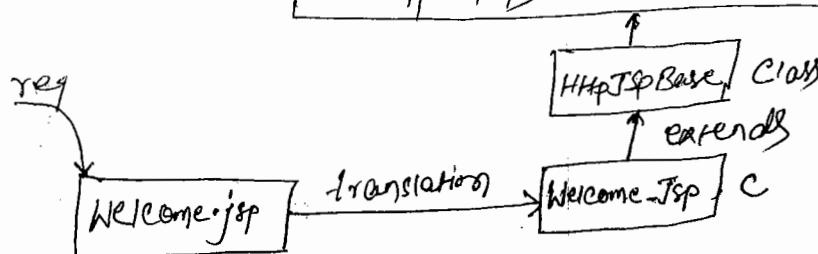
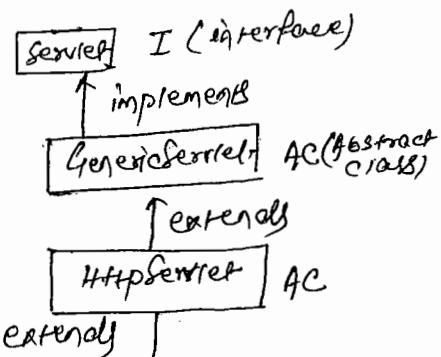
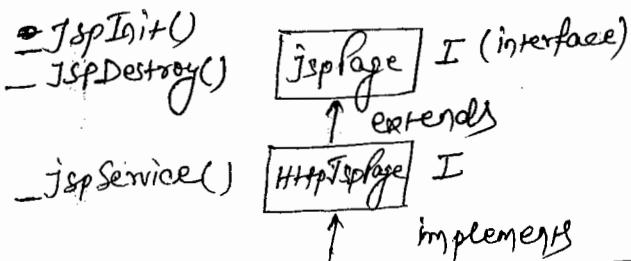
→ In Tomcat Server, the translated service is available in the following location.

C:\Tomcat 8.0\work\Catalina\localhost\firstjspapp\org\apache\jsp\welcome.jsp.java

→ If the jsp file name is welcome.jsp then the translated service class name is Welcome.jsp.

→ All the translated service classes are provided by Tomcat Server as "public and final".

→ The default super class for the translated service class is "HttpJspBase".



## ④ Servlet Compilation:-

Container will compile the translated Servlet class.

## ⑤ Servlet Loading:-

Container will load Servlet class by recode to the memory.

## ⑥ Servlet Instantiation:-

Container will create object for the loaded Servlet class.

## ⑦ Servlet Initialization:-

Container will execute `JspInit()` method to prepare the required resources in Servlet Initialization.

## ⑧ Creating Request and Response Objects:-

→ After Servlet initialization, Container will create a thread to execute `JspService()` method, here to access `JspService()` method, Container must create `HttpServletRequest` and `HttpServletResponse` objects.

## ⑨ Generating Dynamic Response:-

By executing `JspService()` method, Container will generate the required response in response object.

## ⑩ Dispatching dynamic response to Client:-

→ When Container generated thread reached to the ending point of the `JspService()` method then that thread come to dead state, with this container will dispatch response to main server, where main server will forward response to protocol, where protocol will perform the following actions.

① Protocol will prepare response format with header part and body part, where header part is able to manage metadata of the generated dynamic response and body part is able to manage the actual dynamic response.

② Protocol will carry response format to client.

③ Protocol will terminate the logical connection between Client and Server.

⑪ Destroying Request and Response Object:-

When Connection is terminated between client and server, Container will destroy request and response object.

⑫ Servlet Deinstantiation:-

→ After destroying request and response objects, Container will go to waiting state depending on the server, after the waiting state, if no further request are coming to the same resource then Container will destroy servlet object.

⑬ Servlet Unloading:-

→ Container will remove servlet bytecode from Operational memory.

⑭ JSP Unloading:-

→ Container will remove JSP code from Operational memory.

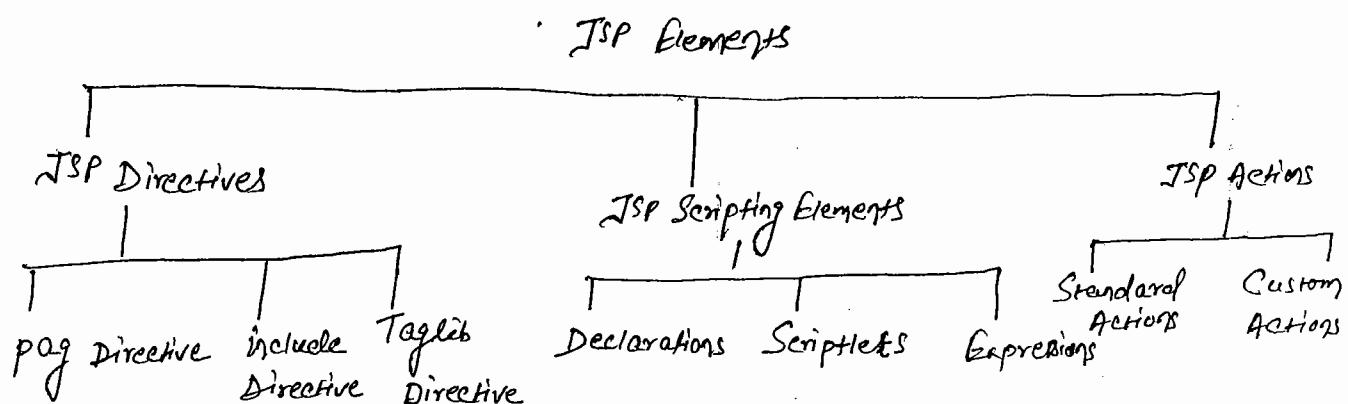
⑮ Destroying ServletContext Object:-

→ Container will destroy ServletContext object when we shutdown the server or when we undeploy the web application.

02/01/2016

## JSP Elements:-

- To prepare JSP pages in web applications we have to use the following elements.



Q. What are the differences between JSP Directives and Scripting Elements?

→ In JSP pages, JSP Directives are used to define preset JSP page properties, including target resource context info present JSP page and to make available user defined tag libraries to the present JSP page.

In JSP page, JSP Scripting elements are used to provide java code.

→ All ~~jsp page~~ <sup>Directives</sup> are recognized and executed at the time of translating JSP page into Servlet [Translation phase].

All the Scripting elements are recognized and executed at the time of request processing.

Note:- In JSP lifecycle, the phases like JSP loading, JSP parsing, JSP Translation to Servlet and Servlet compilation are come under "Translation Phase" and all the remaining JSP Life cycle actions are come under "Request Processing Phase".

⑤ Majority of the JSP Directives are not giving effect to response generation.

Majority of the JSP Scripting Elements are giving effect to response generation.

≡

Q: To prepare JSP pages, we have already Scripting elements then what is the requirement to use JSP Actions?

→ In JSP pages, Scripting elements are used to provide Java code, it is against to JSP rules and regulations. To preserve JSP rules and regulations we have to remove JAVA Code from JSP pages, for this we have to remove Scripting elements from JSP pages. In JSP pages, if want to remove Scripting elements then we have to use an alternative that is "JSP Actions".

In case of JSP Actions, we will remove java code from JSP page, we will define a Scripting tag instead of JAVA Code and we will provide the same amount of JAVA code under class folder.

In this Context, when Container encounter the provided Scripting tag, Container will execute the internally provided JAVA code and Container will perform an action called as JSP Action.

### JSP Directives:-

The main purpose of JSP Directives is

- ① To define present JSP page properties.
- ② To include target resource Content inside the JSP page.
- ③ To make available user defined tag library in the present JSP page.

## SYNTAXES:-

### ① JSP Based Syntax:-

`<%@directive_Name[Attributes_List]%>`

Ex:- `<%@include file = "loginform.html"%>`

### ② XML Based Syntax:-

`<jsp:directive.directive_name[Attribute_List]/>`

Ex:- `<jsp:directive.include file = "loginform.html"/>`

There are three types of JSP directives.

- ① Page directive
- ② Include directive
- ③ Taglib directive

### ① Page directive :-

→ The main purpose of page directive is to define present JSP page properties like specifying a particular language to use inside scripting elements, importing packages, specifying a particular super class to the translated servlet, ...

#### Syntax:-

### ① JSP based Syntax:-

`<%@Page [Attributes_List] %>`

### ② XML based Syntax:-

`<jsp:directive.page [Attributes_List]/>`

→ If we want to prepare page directives in JSP pages then we have to use the following attributes.

- ① language
- ② contentType
- ③ extends
- ④ import
- ⑤ info
- ⑥ Buffer
- ⑦ autoFlush
- ⑧ errorPage
- ⑨ isErrorPage
- ⑩ session
- ⑪ isThreadSafe
- ⑫ isELIgnored

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### ① language:-

→ This attribute will take a particular Scripting Language name to use its coding part inside the Scripting elements in JSP pages.

→ The default value for this attribute is "JAVA".

Ex:-

```
{% Page language = "java" %}
```

Note: All the servers are supporting only java as a language to use its coding part in Scripting elements.



## (2) ContentType:-

→ This attribute will take a particular MIME type to give an information to the client browser about the type of response that present JSP page is generating.

The default value of this attribute is "text/html".

Ex:- `<%@page contentType = "image/jpeg"%>`

## (3) extends:-

→ These attribute can be used to specify a particular Super class to the translated service.

The default value for this attribute is "HttpJspBase".

Ex:- `<%@page extends = "com.degasoft.MyClass"%>`

where com.degasoft.MyClass must be a child class to HttpServlet and it must be an implementation class to JspPage and HttpJspPage interfaces.

## (4) import:-

→ This attribute can be used to specify package name to import in the present jsp page.

The default value for this attribute is java.lang, java.io, javax.servlet, javax.servlet.http, javax.servlet.jsp.

Syntax:-

`<%@page import = "java.io.*"%>`

→ If we want to import more than one package in a single JSP page then we have to use more than one package name along with single "import" attribute or we have to provide multiple no. of "import" attributes.

Ex:-

<%@page import = "java.io.\* , java.util.\* , java.sql.\* "%>

or

<%@page import = "java.io.\*" import="java.util.\*" import="java.sql.\*"%>

Note:- In page directive only import attribute is repeatable attribute  
No other attributes are repeatable.

(5) info:-

→ This attribute can be used to specify some information about the present JSP page.

Note:- If we specify some information as value to "info" attribute then we are able to access that information by using "getInfoInfor()" method.

The default value of this attribute is "Jasper JSP 2.3 Engine".

(6) buffer:-

→ This attribute can be used to specify a particular size to the buffer which is available in TSPWriter.

Note:- In servlets we are able to use PrintWriter to carry response to response object and <sup>in</sup> JSP we are able to use TSPWriter to carry response. PrintWriter is not Buffered Writer, it will reduce performance of the web applications TSPWriter is a BufferedWriter, it will use a buffer internally to store data and it will improve application performance.

→ The default value of this attribute is "8kb".

Ex:- <%@page buffer="52kb"%>

≡

## ⑦ AutoFlush:-

- It is a boolean attribute, it can be used to give an information to the container about to flush or not to flush response to Client when JSPWriter buffer is filled with the response completely.
- If we provide "true" as value to AutoFlush then container will send response to Client automatically when JSPWriter buffer is filled with the response completely.
- If we provide "false" as value to AutoFlush attribute then container will rise an exception like below when JSPWriter buffer is filled with the response completely.  
`org.apache.jasper.JasperException` with the root cause `java.io.IOException: Error: JSP Buffer overflow.`
- The default value of this attribute is "true".

Ex:-

```
<%@page buffer="52KB" autoFlush="true"%>
```

```
<%

```

```
for(int i=0; i<1000000000; i++) {

```

```
out.println("RAMA");

```

```
}
```

```
%>

```

```
</%

```

Note:- Specifying "0KB" as buffer attribute value and "false" as "autoFlush" attribute value is bad combination, where container will rise an exception.

04/01/16

⑧ errorPage:-

→ This attribute will take an error JSP page to execute and to display error messages in user defined format when we are getting an exception in the present jsp page.

Ex:- `<%.page errorPage = "error.jsp"%>`

⑨ isErrorPage:-

→ This attribute is Boolean attribute it will give an information to the container about to allow or not to allow exception implicit object in the present jsp page.

→ if we provide 'true' as value to "isErrorPage" attribute the container will allow exception implicit object to the present JSP page.

→ if we provide "false" as value to "isErrorPage" attribute then container will not provide exception implicit object to the present jsp page.

→ The default value for this attribute is "false".

Ex:- `<%.@page isErrorPage = "true"%>`

Ex:- Welcome.jsp:-

`<%.@ page errorPage = "error.jsp"%>`

`{%>`

`{%<`

`int a=100;`

`int b=0;`

`float =a/b;`

`%>`

`{%<%>`

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Error.jsp:-

```
<%@page isErrorPage = "true"%>
<font color = "red">
<h1>
----- <br>
```

```
<% = exception %><br>
-----
```

```
</h1>
```

```
</font>
```

05/01/16

Ex:- ① Page no - ②) in Material (for employee details)

Ex:- ② Sir will give me handwritten. (for login.jsp) Re-Registration.jsp.

### JSP Actions

#### Session:-

- It is a boolean attribute, it can be used to give an instruction to the Container about to allow or not to allow session implicit object to the present jsp page.
- If we provide true value to "session" attribute then container will allow session implicit object to present jsp page.
- If we provide false value to "session" attribute then Container will not allow session implicit Object to the present jsp page.

Note:- The default value for this attribute is "true".

Ex:- <%@page session = "false"%>

E

### isThreadSafe :-

- It is a boolean attribute, it can be used to give an information to the container about to allow more than one request — not at a time to present JSP page.
- If we provided "true" value to "isThreadSafe" attribute then we are saying to the container that the present JSP page isThreadSafe, so that allow more than one request[Thread] at a time.
- If we provide "false" value to isThreadSafe attribute, then we are saying to the container that the present jsp page is not thread safe, so that allow one request[Thread] at a time.
- The default value for this attribute is "True".

Ex:- <%@page isThreadSafe = "false"

Ex:- <%@page isThreadSafe = "true" %>

<h1>

<%

try {

System.out.println(Thread.currentThread().getName());

Thread.sleep(1000);

System.out.println(Thread.currentThread().getName());

}

Catch (Exception e)

{

e.printStackTrace();

}

<%>

</h1>

%>

### isELIgnored:-

- If it is a Boolean attribute, it can be used to give an instruction to the Container about to allow or not to allow Expression language syntaxes to the present jsp page.
- If we provide "true" value to "isELIgnored" attribute then Container will not allow Expression language Syntaxes.
- If we provide "false" value to "isELIgnored" attribute then Container will allow expression language Syntaxes.
- The default value of this attribute is "false".

Ex:- `<%@page isELIgnored="true"%>`

### include Directive:-

- The directive can be used to include the target resource content into the present jsp page.

#### Syntax:-

`<%@include file = "-----"%>`

- Where "file" attribute will take name and location of the target resource.

#### XML based Syntax:-

`<jsp:directive.include file = "----"/>`

#### includedDirective

```
| → home.jsp  
| → header.htm  
| → footer.htm  
| → welcome.htm  
| → loginform.htm  
| → WEB-INF  
| → Classes ↗
```

## home.jsp

```

<html>
<body>
  <table width="100%" height="100%">
    <tr height="20%">
      <td colspan="2" bgcolor="yellow">
        <%@ include file="header.html" %>
      </td>
    </tr>
    <tr height="65%">
      <td width="60%" bgcolor="lightyellow">
        <%@ include file="welcome.html" %>
      </td>
      <td width="40%" bgcolor="lightblue">
        <%@ include file="loginform.html" %>
      </td>
    </tr>
    <tr height="15%">
      <td colspan="2" bgcolor="blue">
        <%@ include file="footer.html" %>
      </td>
    </tr>
  </table></body></html>
  ≡

```

## header.html

```

<html>
<body>
  <font color="white" size="7">
    <center>
      <b>Durga Software Solutions</b>
    </center>
  </font></body></html>
  ≡

```

## welcome.html

```

<html><body><br><br>
<font color="red" size="6">
  <center><b>
    <marquee>welcome to Durga Software
    Solutions</marquee>
  </b></center>
</font></body></html>
  ≡

```

```

<center><font><body></body></html>
  ≡

```

### loginform.htm :-

```

<html><body><br><br>
<center><table>
<tr><td>User Name</td></tr>
<tr><td><input type="text"/></td></tr>
<tr><td>password</td></td>
<tr><td><input type="password"/></td></tr>
<tr><td><input type="submit" value="login"/>
</td></tr>
</table></center></body></html>

```

### Taglib Directive :-

→ The main purpose of taglib directive is to make available user defined tag library to the present jsp page.

Ex:- `<%@taglib uri="---" prefix="---"%>`

→ Where "uri" attribute will take the name and location of the user defined taglibrary.

→ Where "prefix" attribute will define prefix name to custom tags.

### Scripting Elements :-

→ The main intention of Scripting elements is to allow java code inside the jsp pages.

→ There are three types of Scripting Elements.

- ① Declarations
- ② Scriptlets
- ③ Expression

### ① Declarations :-

→ It able to allow all java declarations like variables, methods, classes, interfaces.

Syntax:- `<%-- Java Declarations --%>`

Note:- If we provide java declaration by using declaration scripting elements, then all the java declarations are provided as cross level declarations in translated Servlet.

### footer.htm

```

<html><body>
<font color="white" size="6">
<center>
<b>Durga Software Solutions,<br>
Ameerpet, Hyderabad </b>
</center>
</font>
</body></html>

```

≡

### Taglib Directive :-

→ The main purpose of taglib directive is to make available user defined tag library to the present jsp page.

Ex:- `<%@taglib uri="---" prefix="---"%>`

→ Where "uri" attribute will take the name and location of the user defined taglibrary.

→ Where "prefix" attribute will define prefix name to custom tags.

### Scripting Elements :-

→ The main intention of Scripting elements is to allow java code inside the jsp pages.

→ There are three types of Scripting Elements.

- ① Declarations
- ② Scriptlets
- ③ Expression

### ① Declarations :-

→ It able to allow all java declarations like variables, methods, classes, interfaces.

Syntax:- `<%-- Java Declarations --%>`

Note:- If we provide java declaration by using declaration scripting elements, then all the java declarations are provided as cross level declarations in translated Servlet.

≡

## Scriptlets:-

→ This Scripting elements can be used to a block of java code inside the JSP pages.

Syntax:- `<% -- Block of java codes -- %>`

Note:- If we provide a block of java code inside Scriptlets then that block of java code will be provided inside `_JSPService(--)` method.

## Expressions:-

→ It can be used to a single expression and displaying the result of that expression on Client browser.

Syntax:- `--> <%= java expression %>`

Note:- If we provide any java expression by using expression Scripting elements Then that expression will be provided to `_JSPService(-,-)` method as parameter to `method are` method.

## ex:- (Scripting elements)

date.jsp

```
<%@page import = "java.util.*"%>
<%
Date d=new Date();
String date=d.toString();
%>
<%
d=new Date();
date=d.toString();
%>
<h1> Today Date :<%=date%>
</h1>
```

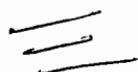
date.jsp.java

```
import java.util.*;
public final class date.jsp extends HttpServlet
{
    Date d=new Date();
    String date="";
    public void _jspService(,)
    {
        d=new Date();
        date=d.toString();
        out.println("Today Date:" + d);
    }
}
```

Z

## JSP Implicit Objects:-

- In web applications, we are able to use some objects like PrintWriter, ServletRequest, ServletResponse, ServletConfig, ServletContext, ... frequently.
- To get the above specified objects in servlets they we have to use some java methods like getServletConfig(), getServletContext(), getSession(), ...
- JSP tech is more developers friendly tech, it has provided all the above frequently used objects as predefined objects called as "JSP Implicit Objects".
- JSP tech has provided the following Implicit Objects along with their referred classes and interfaces.
  - ① out → javax.servlet.jsp.JspWriter
  - ② request → javax.servlet.http.HttpServletRequest
  - ③ response → javax.servlet.http.HttpServletResponse
  - ④ page → java.lang.Object
  - ⑤ config → javax.servlet.ServletConfig
  - ⑥ application → javax.servlet.ServletContext
  - ⑦ session → javax.servlet.http.HttpSession
  - ⑧ exception → java.lang.Throwable
  - ⑨ pageContext → javax.servlet.jsp.PageContext



Q. What are the differences between PrintWriter and JSPWriter?

→ ① PrintWriter is used mainly in Servlets to carry response.  
JSPWriter is used mainly in JSP to carry response.

② PrintWriter is not BufferedWriter.  
JSPWriter is BufferedWriter.

③ PrintWriter will reduce the performance of the web appn.  
JSPWriter will increase the performance of web applications.

Q. What is PageContext and what is the requirement of the PageContext in JSP tech?

→ PageContext is one of the JSP implicit object, it can be used to manipulate data in page scope by using the methods like setAttribute(), findAttribute(), getAttribute(), ... and it can be used to make available all JSP implicit objects in non jsp environment like Tag Handler classes in Custom tags design.

→ To get JSP implicit, PageContext has provided the following methods.

public xxx jeffXXX()

where  $\text{xxx}$  may be `Out`, `Request`, `Response`, ...

public Isplitter getOut()

public HttpServerRequest getHttpRequest()

```
public HttpHeaders getHeaders()
```

public HttpSession getSession()

三

## JSP Scopes:-

- In JAVA applications, to manage the data we are able to use scopes like private, (default), protected and public.
- Similarly, in JSP applications, JSP tech has provided the following four types of scopes to manage the data along with the above specified scopes.

① Page Scope

② Request- Scope

③ Session Scope

④ Application Scope.

① Page Scope:-

→ If we declare data in present JSP page then that data is available upto the present JSP page, this scope is Page Scope.

② Request-Scope:-

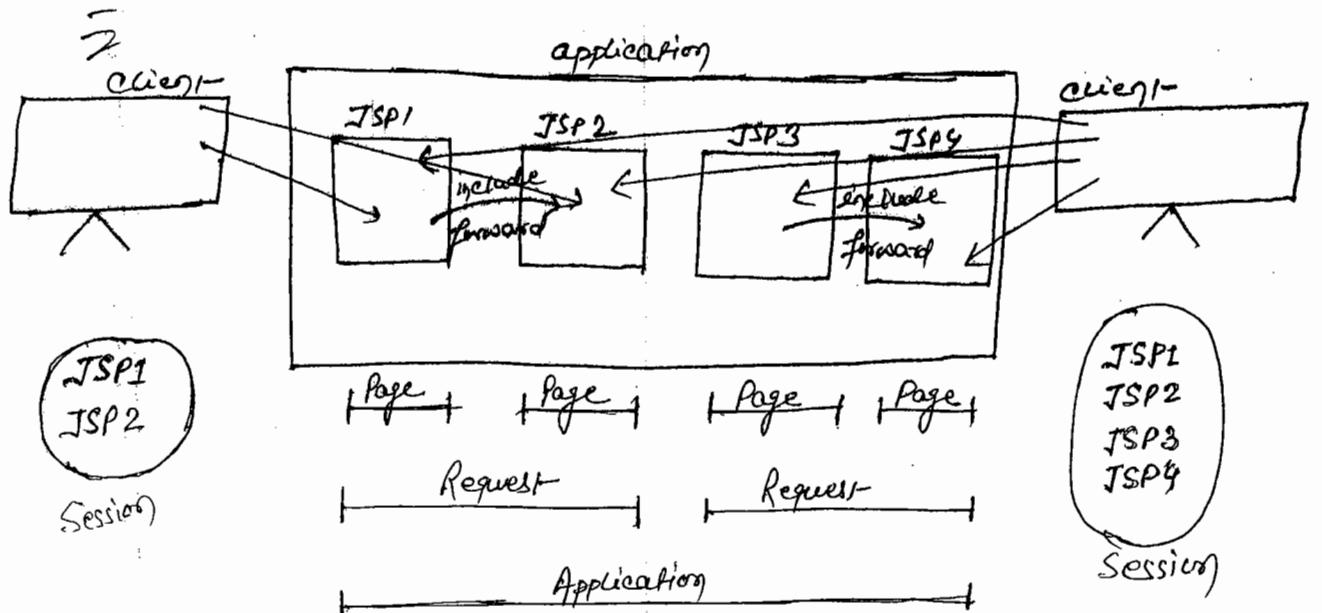
→ If we store data in request object then that data is having request scope, it's available upto all the resources which are visited by the present request object.

③ Session Scope:-

→ If we declare data in HttpSession Object then that data is having session scope, it's available upto all the resources which are visited by the present user.

### ④ Application Scope:-

→ If we store data in `ServletContext Object` then that data is having Application Scope, it's available upto all the resources which are available in the present web application.



Ex:- ① Page 70 - (21) & (22) in Material (for Employee details)

Ex:- 2 Sir will give the handnotes (for Login.jsp)

06/01/16

### JSP Actions:-

User-Registration.jsp

→ In JSP technology, by using Scripting elements, we are able to provide java code inside JSP pages, it is against to the JSP rules and regulations. In this context, to preserve JSP rules and regulations, we have to remove java code from JSP pages, for this we have to remove Scripting Elements from JSP pages. If we want to remove Scripting Elements from JSP pages then we have to use an alternative that is "JSP Actions".

→ In case of JSP actions, we will provide a Scripting tag in JSP pages in place of JAVA code and we will provide the required concept of java code internally w.r.t the Scripting tag. When container encounter Scripting tag in JSP page, Container will execute the corresponding java code which we have provided internally and container will perform an action called as JSP Action.

→ There are two types of actions in JSP.

- ① Standard Actions
- ② Custom Actions.

### ① Standard Actions :-

→ These actions predefined actions provided by JSP tech along with JSP API. JSP has provided all the frequently used actions in web applications as predefined actions.

→ JSP has provided all the predefined actions in the form of a set of tags called as 'Action Tags'.

- ① `<jsp:useBean>`
- ② `<jsp:setProperty>`
- ③ `<jsp:getProperty>`
- ④ `<jsp:forward>`
- ⑤ `<jsp:include>`
- ⑥ `<jsp:param>`
- ⑦ `<jsp:plugin>`
- ⑧ `<jsp:fallback>`
- ⑨ `<jsp:params>`

⑩ <jsp:declaration>

⑪ <jsp:scriptlet>

⑫ <jsp:expression>  
≡

### Java Beans in Java Applications:-

- Java Bean is a reusable component, it is a class, it able to enclose a set of properties and its respective setXXX() and getXXX() methods.
- In general, in Java/J2EE Applications we are able to use Java bean classes to represent entities state/data.
- In web applications, we will use java bean classes to store form data at server side.
- If we want to use java bean classes in Java/J2EE application, then we have to use the following Conventions:

- ① We have to declare an user defined class and it is suggestable to implement java.io.Serializable in order to make eligible java bean object to transfer in network.
- ② Always, it is suggestable to declare java bean classes as public and non-abstract, because we have to create objects for java bean classes.
- ③ In java bean classes, we have to declare properties w.r.t an entity or w.r.t an user form.
- ④ In java bean classes, all the variables must be declared as "private".

⑤ In java bean classes, we have to declare a separate set of `setXXX()` and `getXXX()` methods in order to set data to Java Bean and get data from Java Bean objects.

⑥ In java bean classes, we have to declare all the methods as "public".

Note:- Points 4 and 6 are able to improve Encapsulation in java application.]

⑦ In java bean classes, if we want to declare any constructor then that constructor must be public and 0-arg constructor.

Eg:-

```
public class Student implements java.io.Serializable {
```

```
    private String sid;
    private String Sname;
    private String Saddr;
    public void setSid(String sid){
        this.sid = sid;
    }
```

```
    public void setSname(String sname){
        this.sid = sid; this.sname = sname;
    }
```

```
    public void setSaddr(String saddr){
        this.saddr = saddr;
    }
```

```

public String getSid(){
    return sid;
}

public String getName(){
    return name;
}

public String getSalary(){
    return salary;
}

```

### ① <jsp:useBean>

→ This tag can be used to interact with java bean object from JSP page.

→ This tag is able to create object for java bean class with the default data.

Syntax:-

`<jsp:useBean id="--" class="--" type="--" scope="--">`

`--`  
`</jsp:useBean>`

→ Where "id" attribute will take a variable to store the generated bean object reference.

→ Where "class" attribute will take fully qualified name of the bean class to create its object.

→ Where "type" attribute will take bean object type that is fully qualified name of bean class, it is not required if we use class attribute e.g. `<jsp:useBean>tag`.

→ Where "Scope" attribute will take a particular JSP scope to store the generated Bean object reference, always it is suggested to provide either Session Scope or application Scope to Scope attribute.

## ② {jsp:setProperty}

→ This tag can be used to set a value to the Bean property by executing the respective SetXXX() method.

Syntax:-

`{jsp:setProperty name="--" property="--" value="--"}`

→ Where "name" attribute will take a reference variable containing Bean object reference.

Note:- Name attribute value in {jsp:setProperty} tag and id attribute value in {jsp:useBean} tag must be same.

→ Where "property" attribute will take a Bean property in order to execute its SetXXX() method.

→ Where "value" attribute will take a value to pass as parameter to the respective SetXXX() method at the time accessing SetXXX() method.

## ③ {jsp:getProperty}

→ This tag can be used to get a particular Bean property value by executing its GetXXX() method.

Syntax:-

cont..

Syntax:-

```
<jsp:setProperty name="..." property="--"/>
```

→ Where "name" attribute will take a reference variable contains bean object reference.

[Note:- id attribute value in `<jsp:useBean>` tag and name attribute value in `<jsp:setProperty>` tag must be same, Where "property" attribute will take a bean property name to execute its `getXXX()` method.

[Note:- `<jsp:setProperty>` tag and `<jsp:getProperty>` tag must be provided as child tags to `<jsp:useBean>` tag.]

[Note:- If all form property name and java bean property names are same then it is possible to copy all request parameter values from request to the respective java bean object directly by providing "property" attribute value "\*".

```
<jsp:useBean id="u" class="User" type="User" public class User{  
    Scope = "session"} private uname, upwd;
```

```
<jsp:setProperty name="u" property = "uname" public void setUname(String uname){  
    value = "AAA"/> } this.uname = uname;
```

```
<jsp:setProperty name="u" property = "upwd" public void setUpwd(String upwd){  
    value = "abc123"/> } this.upwd = upwd;
```

UNAME: <jsp:getProperty name = "u"  
 property = "uname"/> public String getUname(){  
 return Uname; }

UPWD: <jsp:getProperty name = "u"  
 property = "upwd"/> public String getUpwd(){  
 return upwd; }

```
</jsp:useBean>
```

≡.

Usebeanapp

| → employeeform.htm, display.jsp

| → WEB-INF

|  
| → classes.

| → Employee.java

| → com.durgsoft.Employee.class

program :- page-26 (Application-22)

### <jsp:include>

→ The main intention of this action tag is to include the response of the target resource into the present JSP page response.

Syntax :-

<jsp:include page = " --- " flush = " --- "/>

→ Where "page" attribute will take the name and location of the target resource.

→ Where "flush" attribute is a boolean attribute, it can be used to flush the response to client when buffer is filled with the response.

Q. What are the differences between include directive and include action tag?

→ include directive can be used to include target resource content into the present JSP page.

include action tag can be used to include the target resource response into the present JSP page.

→ In web application, we will use include directive to include static resources where frequent updates

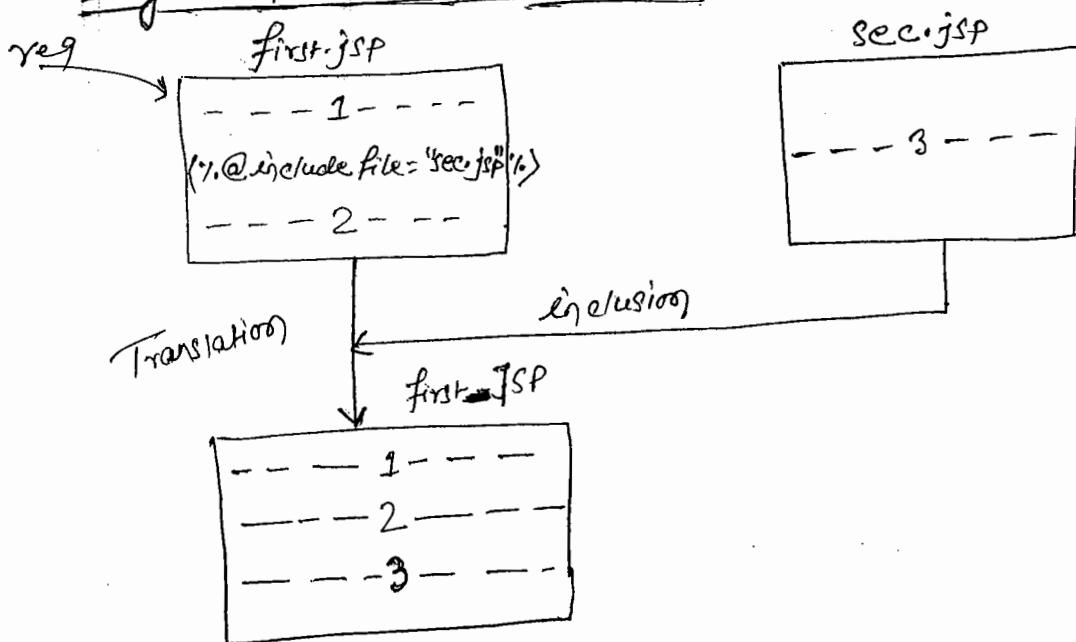
are not available:

In web application, we will use include action tag to include dynamic resources where frequent updations are available.

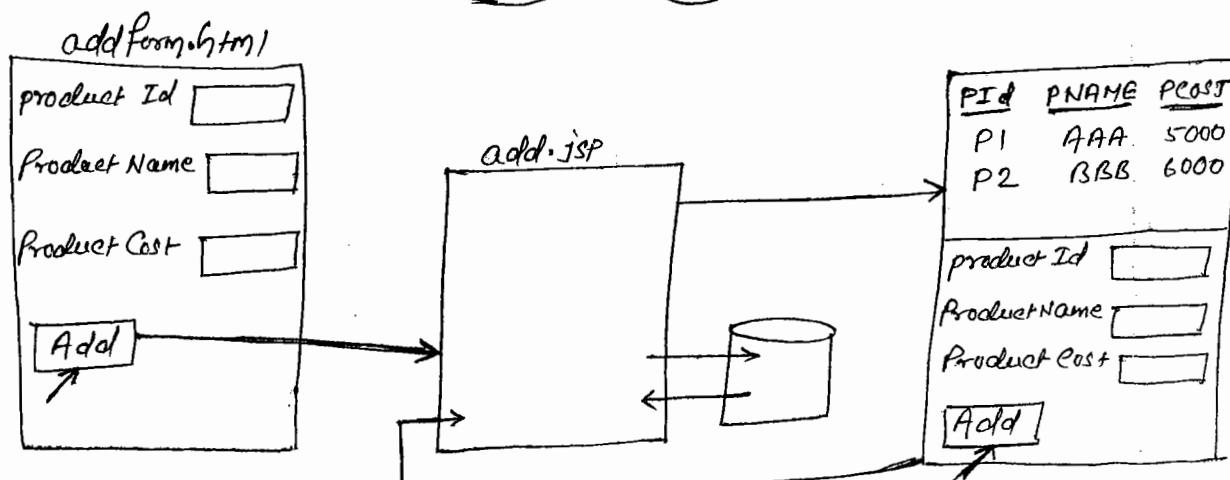
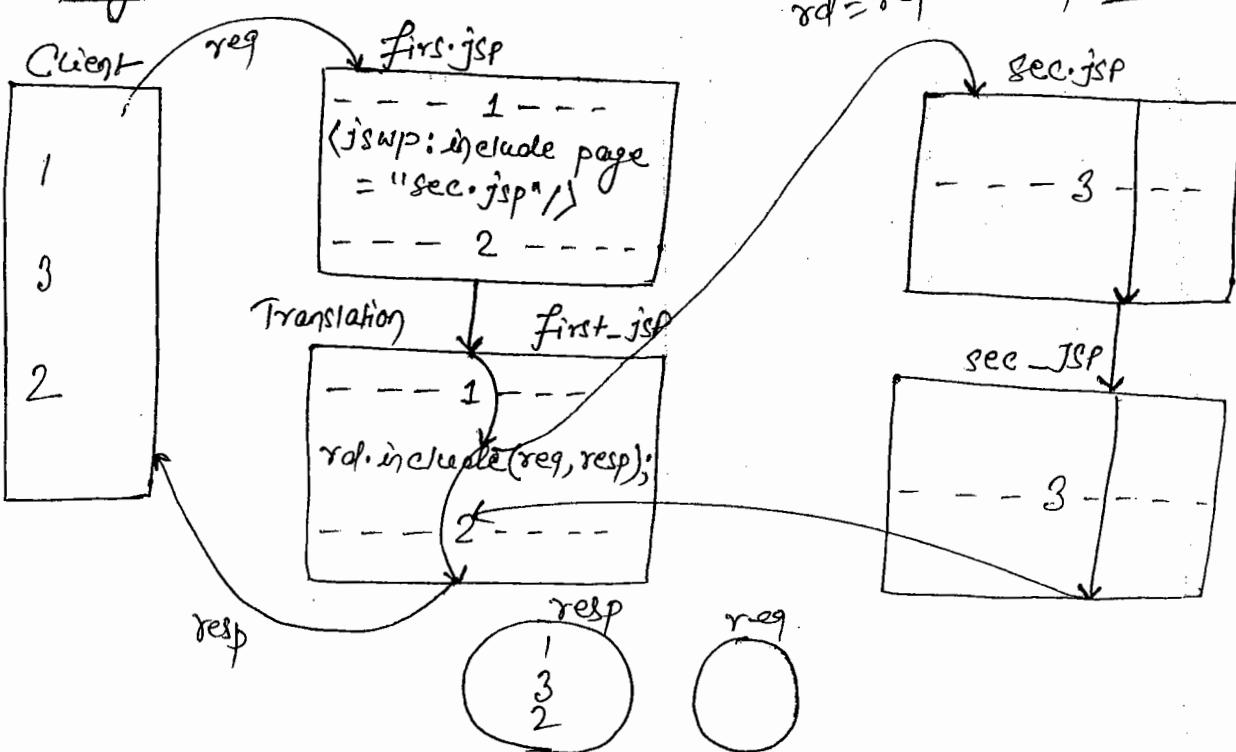
- include directive is basically a directive, it will be executed at the time of translating jsp page into servlet.
- include action tag is basically a jsp action tag. It will be executed at the time of request processing.
- include directive will provide static inclusion of data.
- include action ~~tag~~ tag will provide dynamic inclusion of the data.
- If we include a jsp page content into another jsp page by using include directive then Container will create only one translated servlet.

If we include a jsp page into another jsp page, by using include action tag then Container will create two translated Servlet.

### Diagram for include directive:-



## Diagram for include action



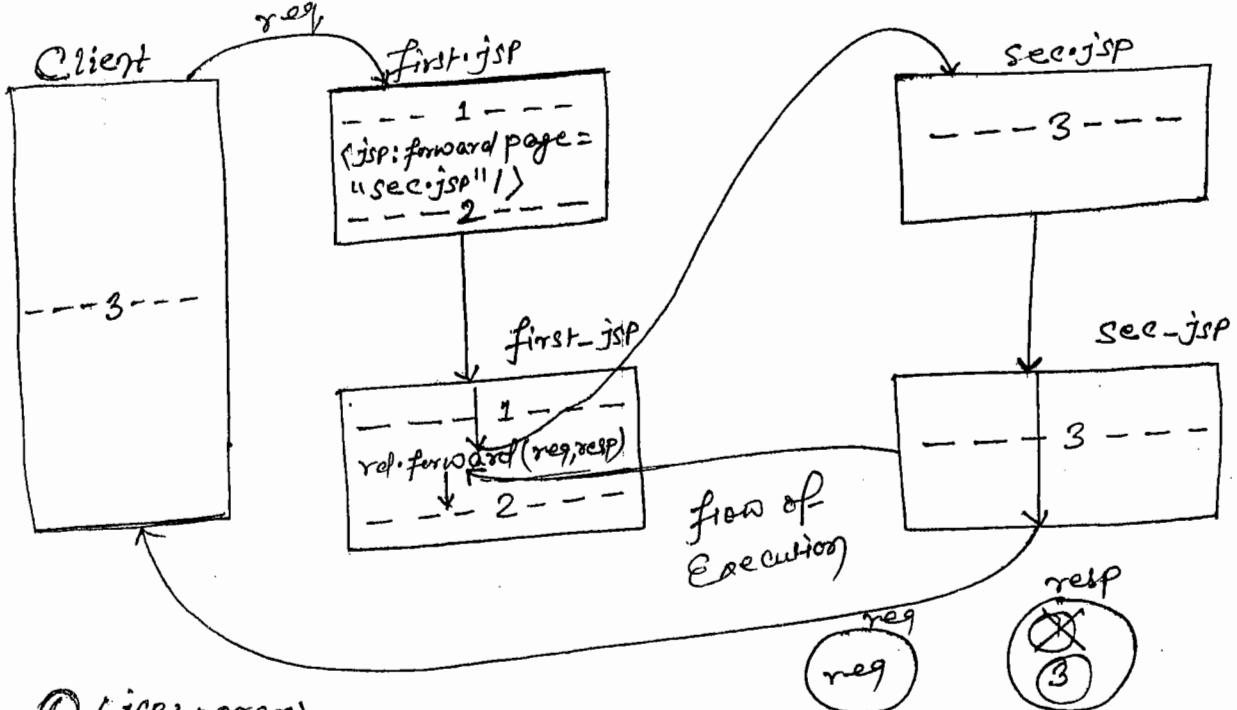
### ⑤ <jsp:forward>

→ This action tag can be used to forward request to a particular target resource from present JSP page.

Syntax:-

`<jsp:forward page = " --- "/>`

→ Where page attribute will provide name and location of the target resource.



#### ⑥ <jsp:param>

→ This tag can be used to store a name value pair in request object while performing include action or forward action.

Syntax:- <jsp:param name="---" value="---"/>

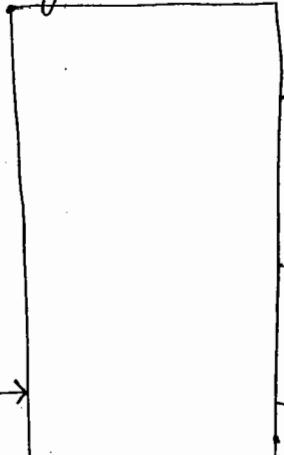
→ Where "name" and "value" attributes are providing name and value in request parameter which we want to store in req. object.

[Note:- <jsp:param> tag must be used as nested tag to <jsp:include> and <jsp:forward>]

registrationform.htm

Uname	Sweta
Vage	24
Vaddr	Hyd
<input type="button" value="Registration"/>	

registration.jsp



Existed.htm

User Existed  
already.

Success.htm

Registration  
Success

Failure.htm

Registration  
Failure

## <jsp:plugin>

→ The main purpose of this tag is to include an applet into the present JSP page.

### Syntax:-

```
<jsp:plugin code="---" width="---" height="---">  
</jsp:plugin>.
```

→ Where "code" attribute will take full qualified name of the applet.

→ Where "width" and "height" attributes will take applet size.

## <jsp:fallback>

→ The main purpose of this tag is to display an alternative message in the respective applet if it's not loaded and initialized.

### Syntax:-

```
<jsp:fallback>  
--- description ---  
</jsp:fallback>
```

## <jsp:params>

→ In case of <jsp:plugin> tag, it is \* to + initialization parameters to the respective applet from jsp page, where to represent a single initialization parameter we have to use <jsp:param> tag.

→ If we want to provide more than one <jsp:param> tags then all the <jsp:param> tags must be enclosed with <jsp:params> tag.

✓

Syntax:-

<jsp: params>

<jsp: param name = " --- " value = " --- "/>

<jsp: param name = " --- " value = " --- "/>

<jsp: param name = " --- " value = " --- "/>

-----

</jsp: params>

<jsp: declaration>

→ This tag is same as declaration Scripting elements, it able to allow all java declarations like variables, methods, ... etc.

Syntax:-

<jsp: declaration>

--- java declarations ---

</jsp: declaration>

<jsp: scriptlet>

→ This tag is same as Scriptlet scripting elements, it able to allow a block of java code.

Syntax:-

<jsp: scriptlet>

--- block of java code ---

</jsp: scriptlet>

<jsp: expression>

→ This tag is same as expression Scripting elements. it able to allow a single java expression and it will display the result of the specified expression.

Syntax:-

<jsp: expression>

--- java expression ---

</jsp: expression>

Ex:- date.jsp

<%@page import="java.util.\*"%>

<jsp: declaration>

{}

## Custom Actions:-

- In jsp page, to remove java code and scripting elements, we have to use jsp Actions. In jsp actions, we are able to use standard actions to remove java code, but standard actions are having limited no. of actions and having bounded functionalities, so that standard actions are not sufficient to remove java code from jsp pages completely.
- In the above context, if we want to remove java code completely from jsp pages, then we have to use Custom Actions.
- Custom actions are the jsp actions defined by the developers as per their application requirement.
- In jsp applications, developers will provide Custom Actions by defining a set of user defined tags called as custom tags.

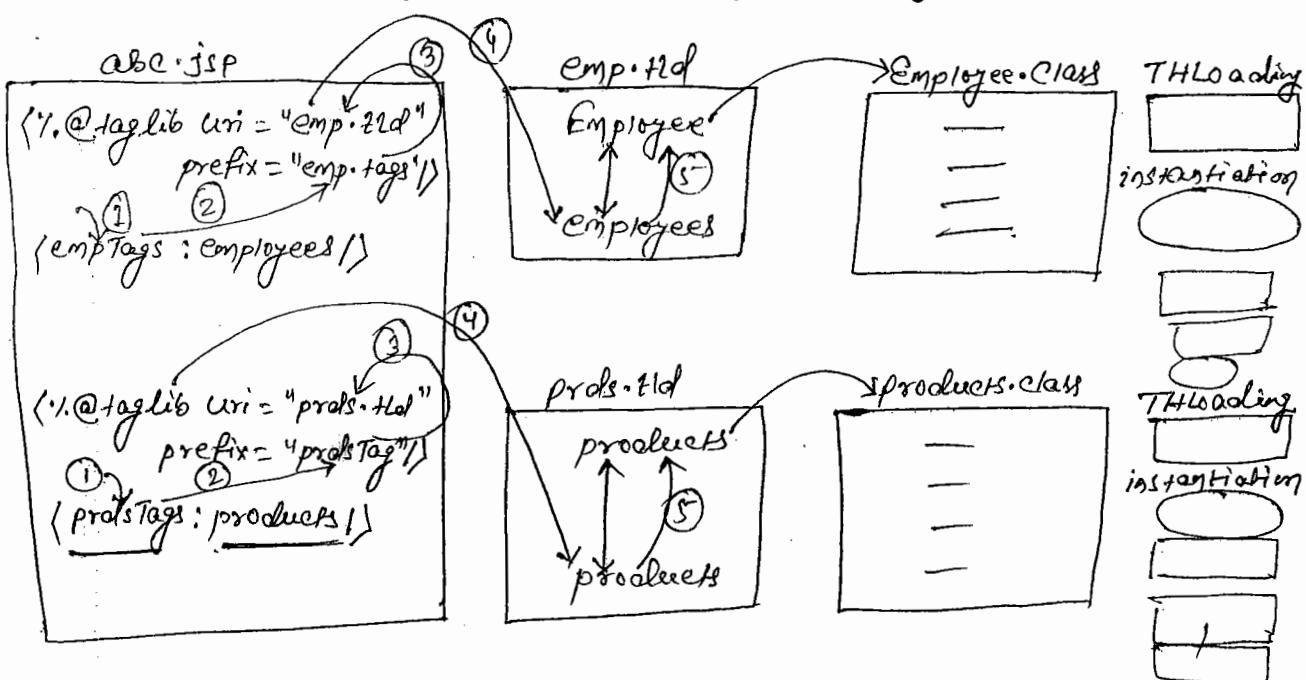
Syntax:-

<prefix-Name:tag-Name[Attribute-list]> → Start tag.  
                    
                   Body.

</prefix-Name:tag-Name> → End tag.

- To prepare custom tags in jsp application we have to use the following elements.
  - ① JSP page with taglib directory
  - ② TLD file
  - ③ Tag Handler class.
- where taglib directive is able to specify the name and location of the tld file on the basis of the prefix names of the custom tags.
- When tld file can be used to provide the mapping between custom tag name and the respective tag handler class.

- Where tag Handler class are providing functionalities of the custom tags.
- When Container encounter custom tags in jsp page then Container will perform the following actions.
  - Container will read prefix name and custom tag name.
  - Container will identify a taglib directive on the basis of the prefix attribute value.
  - Container will get "uri" attribute value from respective taglib directive that is the same and location of tld file.
  - Container will goto the respective tld file and identify the respective tag handler class.
  - After getting name and location of the tag handler class, Container will recognize the tag handler class and Container will execute tag handler class by following lifecycle actions.



## ① A JSP page with taglib directory:-

→ The main purpose of taglib directive is to specify the name and location of tld file and to define prefix name to the custom tags.

```
<%@taglib uri="---" prefix="---"%>
```

→ Where "uri" attribute will take the name and location of tld file.

→ Where "prefix" attribute will take the name of the custom tag.

## ② TLD [Tag Library Descriptor] file:-

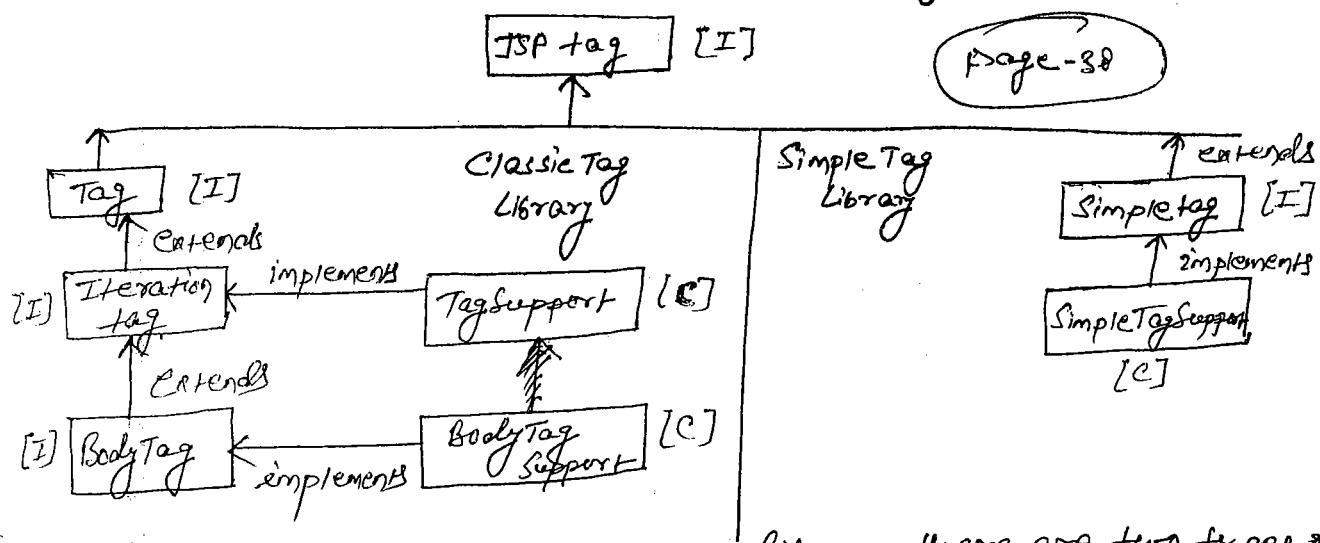
→ The main purpose of "tld" file in JSP application is to provide mapping between custom tag names and the respective tag handler classes, Custom tags attributes description, Expression language functions.

→ To provide mapping between custom tag names and tag handler classes in tld filename, we have to use the following tags.

```
<jsp-version> JSP version </jsp-version>
<taglib-version> tld file version </taglib-version>
<tag>
  <name>Custom tagname </name>
  <tag-class> Fully qualified name of Tag Handler </tag-class>
  <body-content> empty or jsp </body-content>
</tag>
-----
</taglib>
```

## Tag Handler Classes:-

- The main purpose of Tag Handler classes is to define basic functionality of the custom tags.
- In custom tags applications, to prepare tag Handler classes JSP has provided the following predefined classes and interfaces as part of "javax.servlet.jsp.tagext" package.



→ As per the above custom tag library, there are two types of custom tags.

- ① Classic tags
- ② Simple tags.

### Classic Tags:-

→ As per the classic tags library, there are 3 types of custom tags.

- a) Simple classic tags
- b) Iterator tags
- c) Body tags.

## ① Simple Classic Tag:-

- If any classic tag is defined without body and attributes, then that custom tag is called simple classic tag.
- If we want to prepare simple classic tag, then the respective tag handler class must implement

javax.servlet.jsp.tagext.Tag interface

public interface Tag extends JSPTag.

{

public static final int EVAL\_BODY\_INCLUDE;

public static final int SKIP\_BODY;

public static final int EVAL\_PAGE;

public static final int SKIP\_PAGE;

public void setPageContent(PageContent pageContent);

public void setParent(Tag t);

public ~~Tag~~ getParent();

public int doStartTag() throws JSPException;

public int doEndTag() throws JSPException;

public void release();

}

public class MyHandler implements Tag

{

— implementation for all the TagInterface methods —

}

- In JSP applications, when container encounters custom tag, then container will execute the following methods internally.

09/01/2016

① setPageContext(-)

- Where "setPageContext(-)" method can be used to inject pageContext JSP implicit object into the present tag handler class, which would be created by the container automatically when custom tag is encountered in jsp page.

② setParent():

- Where "setParent(-)" method can be used to inject parent tag's tag handler class object into the present tag's tag handler class, which could be created by the container automatically after executing setPageContext() method.

③ getParent():

- Where "getParent()" method can be used to get parent tag's tag handler class object if the present custom tag is child tag to some other parent tag.

④ doStartTag():

- Where "doStartTag()" method would be executed when container encounters start tag of the custom tag in order to perform any action w.r.t. the start tag.

→ Executing or not executing custom tag body is completely depending

on the return value of doStartTag() method.

→ There are two possible return values from doStartTag() method.

① EVAL-BODY-INCLUDE

② SKIP-BODY.

→ If we return EVAL-BODY-INCLUDE constant from doStartTag() method then container will execute custom tag body.

→ If we return SKIP-BODY constant from doStartTag() method then container will not execute custom tag body, Container will encounter end tag of custom tag directly.

### ⑤ int doEndTag():-

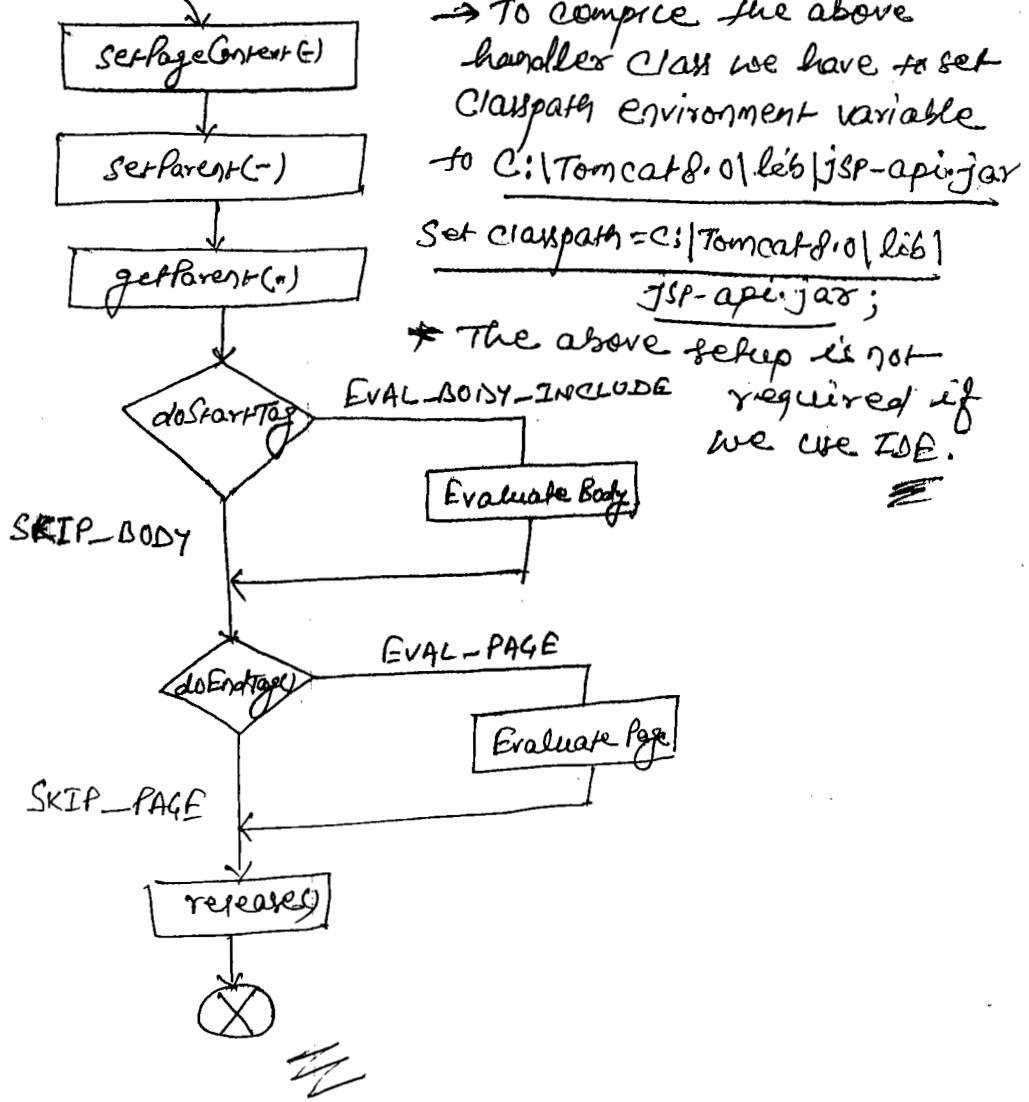
- Where "doEndTag()" method would be executed by the container when container encounters end tag of the custom tag in order to perform an action w.r.t the end tag.
- Executing remaining jsp page after the custom tag or not is completely depending on the return value from doEndTag().
- There are two possible return values from doEndTag() method
  - ① EVAL-PAGE
  - ② SKIP-PAGE.
- If we return EVAL-PAGE Constant from doEndTag() method then container will execute remaining jsp page after the custom tag.
- If we return SKIP-PAGE Constant from doEndTag() method then container will not execute the remaining jsp page after the custom tag.
- Where "release()" method would be executed by the Container when container destroy tag handler class object in order to perform cleanup operation which deinstantiation tag handler class object.

### 6 void release():-

case 1:- If we provide "empty" as value to <body-content> in tld file and if we provide body to the custom tag in jsp page then Container will rise an exception like :-

org.apache.jasper.JasperException:/hello.jsp (line: 2, column: 0) According to TLD, tag mytags:hello must be empty, but it's not.

→ This method will destroy tag handler class object.



→ To compile the above handler class we have to set Classpath environment variable to C:\Tomcat8.0\lib\jsp-api.jar

Set classpath=C:\Tomcat8.0\lib\jsp-api.jar;

\* The above setup is not required if we use IDE.

### Cave-2:-

→ If we provide "jsp" as value to `(body-content)tag` it will fire and if we provide body to the custom tag but if we return "SKIP-BODY" Constant from `doStartTag()` method from tag handler class then container will not rise any exception and at the same time customer will not execute custom tag body.

## \* Attributes in Custom tag:-

→ If we want to provide attributes in custom tags then we have to use the following steps.

(1) In JSP page, provide attribute with a value in custom tag.

```
<mytag:Hello name="Durga"/>
```

(2) In tld file, configure attribute with the following tags:

```
<taglib>
```

```
- - -<tags>
```

```
- - -<attribute>
```

```
<name> attribute name</name>
```

```
<required> true/false</required>
```

```
<rtepxvalue> true/false</rtepxvalue>
```

```
</attribute>
```

```
- - -<tag>
```

```
- - -</taglib>
```

→ Where "<attribute>" tag will represent an attribute. where "<name>" tag will take attribute name.

→ Where "<required>" tag will take true or false values in order to make attribute is mandatory or optional.

→ Where "<rtepxvalue>" tag will take true or false values in order to take runtime expression values.

Eg:-

```
<taglib>
```

```
<jsp-version>2.3</
```

```
<+Lib-version>1.0</
```

```

<tag>
  <name>hello</
  <tag-class>com.durgsoft.HelloHandler</
  <body-content>empty<
<attribute>
  <name>name</
  <required>true</
  <respvalue>true</
</attribute>
</tag>
</tags>
  =

```

- ③ In tag-Handler class, declare a property with the same name of the attribute and declare setXXX() method in order to inject attribute value in the tag handler class.

```

public class HelloHandler implements Tag {
    -----
    private String name;
    public void setname(String name) {
        this.name=name;
    }
    -----
}

```

=====

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## ② Iterator tags:-

- These tags are able to allow to execute custom tag body repeatedly on the basis of a particular condition.
- To prepare iterator tags in JSP applications, the respective tag handler class must implement →  
`javax.servlet.jsp.tagext.IterationTag interface.`

public interface IterationTag extends Tag {

    public static final int EVAL\_BODY\_AGAIN;

    public int doAfterBody() throws JSPException;

}

public class MyHandler implements IterationTag {

    — implementation to all the methods of Tag interface and IterationTag interface —

- To prepare Iterator tags in JSP applications we have to follow the following conditions.

- ① We must provide body to custom tag.
  - ② We must provide "jsp" as value to <body-content> tag in tag handler class.
  - ③ We must set "EVAL\_BODY\_INCLUDE" constant from doStartTag() method in the respective tag handler class.
- If we return EVAL\_BODY\_INCLUDE constant from doStartTag() method then container will execute custom tag body at the end of the custom tag body execution and before getting end tag of the custom tag container will execute doAfterBody() method.

→ Executing custom tag body again or skipping custom tag body is completely depending on the return value of doAfterBody() method.

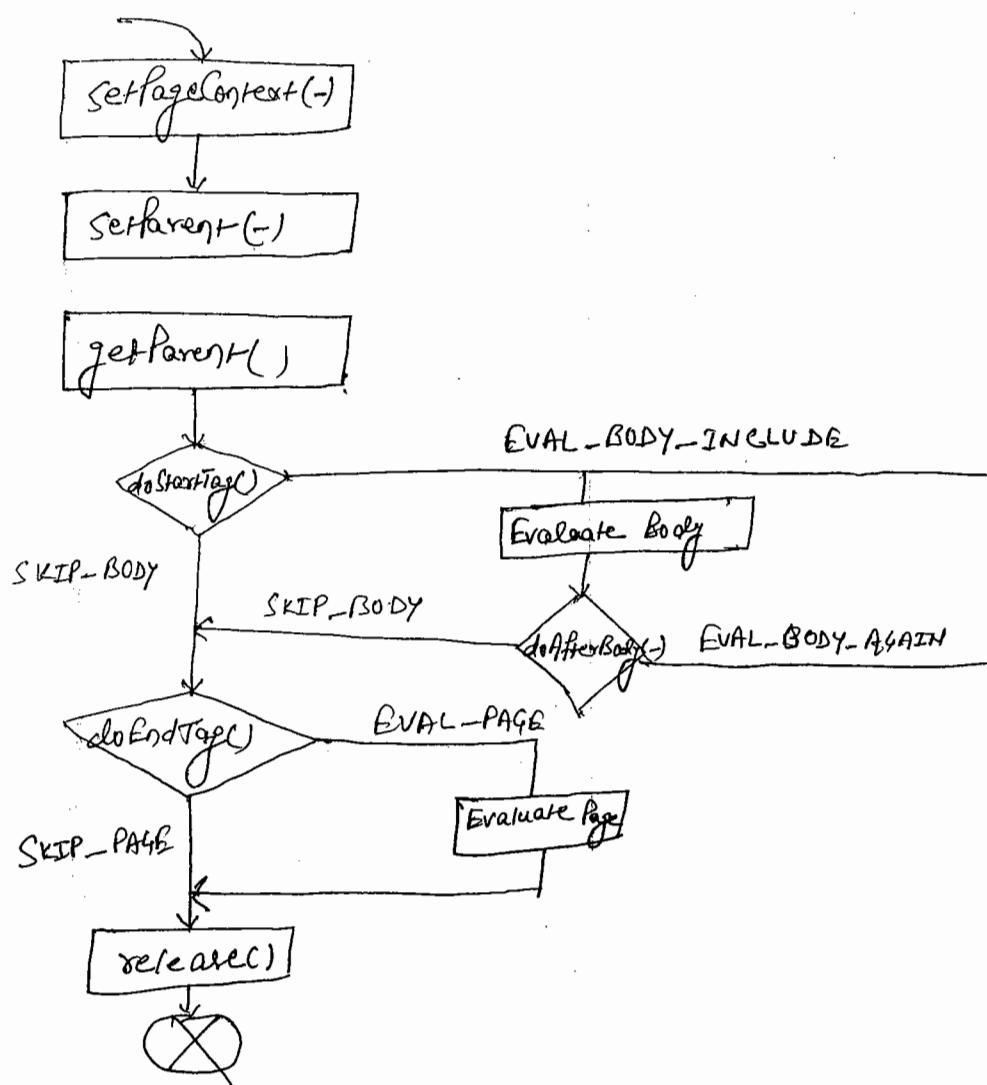
→ There are two possible return values from doAfterBody() method

① EVAL-BODY-AGAIN

② SKIP-BODY

→ If we return EVAL-BODY-AGAIN constant from doAfterBody() method then Container will execute custom tag body one more time.

→ If we return SKIP-BODY constant from doAfterBody() then container will skip custom tag body execution and encounter end tag of the custom tag.



### Conclusion:-

- To prepare custom tags, if we use the above approach that is implementing IterationTag interface in tag handler classes then we have to provide implementation for all the methods of IterationTag interface irrespective of the actual requirements, this approach increase unnecessary methods in tag handler classes.
- To overcome the above problem, JSP API has provided solution in the form of Adapter class that is javax.servlet.jsp.tagext.tagSupport class.
- If we want to prepare custom tags with tagSupport class, then we have to an userdefined class [tag handler class] and it must be extended from tagSupport class, where we have to override required methods instead of implementing all the methods of IterationTag interface.

```
public class TagSupport implements IterationTag {  
    public static final int EVAL_BODY_INCLUDE;  
    public static final int SKIP_BODY;  
    public static final int EVAL_PAGE;  
    public static final int SKIP_PAGE;  
    public static final int EVAL_BODY_AGAIN;  
  
    public PageContent pageContent;  
    public Tag t;  
    public void setPageContent(PageContent pageContent)  
    {  
        this.pageContent = pageContent;  
    }  
}
```

```
public void setParent(Tag t)
{
    this.t = t;
}

public Tag getParent()
{
    return t;
}

public int doStartTag() throws JSPException
{
    return SKIP_BODY;
}

public int doAfterBody() throws JSPException
{
    return SKIP_BODY;
}

public int doEndTag() throws JSPException
{
    return EVAL_PAGE;
}

public void release()
{
}

public class MyHandler extends TagSupport
{
    — Override any required methods —
}
```

page No:- 47  
50

Z

## Nested Tags:-

→ Declaring a tag inside a tag is called as nested tag.

In JSP applications, if we prepare nested tags then we have to provide separate tag configuration for both outer tag and nested tag in the tld file and separate tag handler classes must be provided for both outer tag and nested tag.

Ex:- page NO - 48

## Body Tags:-

- In general custom tags, if we provide body then container will copy the custom tag body and container will display that custom tag body without performing modifications.
- In custom tag applications, if we want to perform modification over custom tag body then we have to use Body tags.
- If we want to prepare Body tags then the respective tag handler class must implement javax.servlet.jsp.tagext.BodyTag interface.

public interface BodyTag extends IterationTag

```
{  
    public static final int EVAL_BODY_BUFFERED;  
    public void setBodyContent(BodyContent bodyContent);  
    public void doInitBody();  
}
```

Cont...

public class MyHandler implements BodyTag

— implementation for all the methods of BodyTag —

→ In case of body tags, there are three return values from doStartTag() method.

- ① EVAL-BODY-INCLUDE
- ② SKIP-BODY
- ③ EVAL-BODY-BUFFERED

→ If we return EVAL-BODY-BUFFERED constant from doStartTag() method then container will get custom tag body in the form of BodyContent object and container will access setBodyContent() method in order to inject BodyContent object to tag handler class and container will access doInitBody() method, in order to prepare BodyContent object for manipulations.

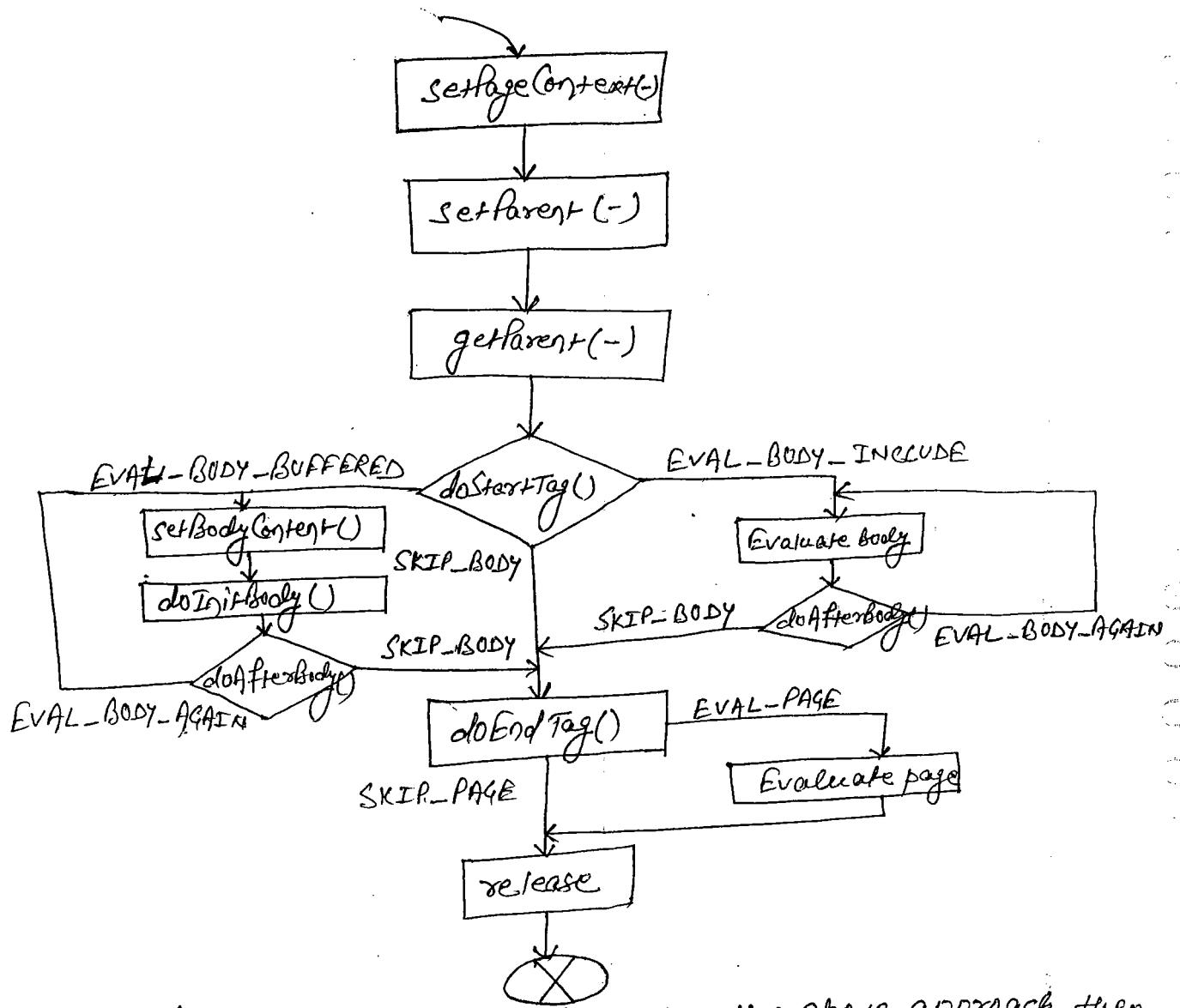
→ To get Custom tag body from BodyContent Object we have to use the following method.

public String getString()

→ To get JSPWriter object in order to submit response to Client we have to use the following method from BodyContent.

public JSPWriter getEnclosingWriter()

Cont...



- To prepare custom tags, if we use the above approach, then we have to implement Bodytag interface in TagHandler class, where we have to provide implementation all the methods of Bodytag interface irrespective of the actual requirement.
- To Overcome the above problem, JSP API has provided an alternative in the form of `javax.servlet.jsp.tagext.BodyTagSupport` class as an adapter class.
- If we want to prepare customtags with `BodyTagSupport` class then the respective tag handler class must be extended from `BodyTagSupport` class, where we have to ~~required~~ override only the required methods, instead of implementing all `BodyTag` interface methods.

```
public class BodyTagSupport extends TagSupport implements BodyTag
{
    public static final int EVAL_BODY_INCLUDE;
    public static final int SKIP_BODY;
    public static final int EVAL_BODY_AGAIN;
    public static final int EVAL_BODY_BUFFERED;
    public static final int EVAL_PAGE;
    public static final int SKIP_PAGE;

    public PageContent pageContent;
    public Tag t;
    public BodyContent bodyContent;

    public void setPageContent(PageContent pageContent)
    {
        this.pageContent = pageContent;
    }

    public void setParent(Tag t)
    {
        this.t = t;
    }

    public Tag getParent()
    {
        return t;
    }

    public void setBodyContent(BodyContent bodyContent)
    {
        this.bodyContent = bodyContent;
    }

    public void doInitBody()
    {
    }

    public int doStartTag() throws JspException
    {
        return EVAL_BODY_BUFFERED;
    }

    public int doAfterBody() throws JspException
    {
        return SKIP_BODY;
    }

    public int doEndTag() throws JspException
    {
        return SKIP_PAGE;
    }
}
```

```
public void release()
{
}
}

public MyHandler extends BodyTagSupport()
{
    — Override the only required method —
}
```

## Simple Tags

- Q: To design custom tags we have already classic tags  
↳ Then what is the requirement to go for simple tags?
  - Classic tags are providing difficult approach to design custom tags.
  - Simple tags are providing simple approach to design simple tags.
  - Classic tags are more API dependent, so that it is very difficult to perform debugging and testing.
    - Simple tags are less API dependent, so that it is very simple to perform debugging and testing.
  - In case of classic tags we have to manage three life-cycle to design custom tags.
    - In case of simple tags, we have to manage one life-cycle to design custom tags.
  - In case of classic tags, by default all the TagHandler class objects are cached objects.
    - In case of simple tags all the tag handler class objects are not cached objects.

— cont... —

NOTE:- Cached objects are objects, which are stored in cache memory to reuse objects in future.

→ In case of classic tags, by default all the custom tags are not body tags.

In case of simple tags, by default all the custom tags are having body tags power.

→ If we want to prepare custom tags by using simple Tag library then the respective tag handler class must implement

javax.servlet.jsp.tagext.SimpleTag interface

public interface SimpleTag extends JspTag

public void setJspContent(JspContent jspContent);

public void setParent(JspTag jt);

public JspTag getParent();

public void setJspBody(JspFragment jf);

public void doTag() throws JspException, IOException;

public MyHandler implements SimpleTag

— implementation for all the Tag interface method —

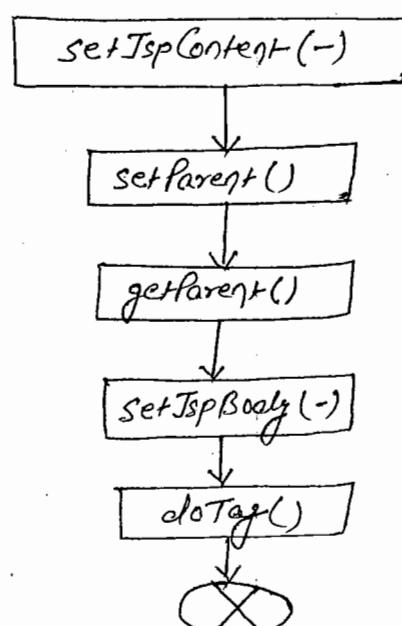
→ Where setJspContent() method can be used to inject JspContent object which is same as PageContent object in classic tags.

→ Where setParent(-) method can be used to inject parent tags tag handler class objects into the present tag handler class.

→ Where getParent() method can be used to get parent tags tag handler class objects.

→ Where setJspBody() method can be used to JspFragment object in the tag handler class which is same as BodyContent object in classic tags.

- Where `doTag()` method will include actual application logic which we want to execute w.r.t the custom tag, which is equivalent to `doStartTag()` method and `doEndTag()` methods in classic tags.
- When container encounter simple tag in jsp page, then container will execute the respective tag handler class by using the following life cycle actions.



- If we use above approach to prepare custom tags, then we must implement all the `SimpleTag` interface methods, irrespective of the actual requirement. In JSP's content to remove unnecessary methods implementations JSP has provided solution in the form of `SimpleTagSupport` as an adaptor class.
- If we want to prepare Custom tags with `SimpleTagSupport` class, then the respective tag handler class must be extended from `SimpleTagSupport` class, where we have to override only few required methods.

3

```
public class SimpleTagSupport implements SimpleTag
{
    public JspContent jspContent;
    public JspContent jt;
    public JspFragment jf;
    public void setJspContent(JspContent jspContent)
    {
        this.jt this.jspContent = jspContent;
    }
    public void setParent(SimpleTag jt)
    {
        this.jt = jt;
    }
    public void setJspBody(JspFragment jf)
    {
        this.jf = jf;
    }
    public JspContent getJspContent()
    {
        return jspContent;
    }
    public SimpleTag getParent()
    {
        return jt;
    }
    public JspFragment getJspBody()
    {
        return jf;
    }
    public void doTag()
    {
    }
}
```

public class MyHandler extends SimpleTagSupport

— Override the only required method —

My

## JSTL (JSP Standard Tag Library)

- In jsp, by using scripting elements, we are able to provide java code inside jsp pages, but it is against to jsp rules and regulations, to prevent jsp rules and regulations we have to remove scripting elements, for this we have to use JSP actions as an alternative.
- To remove java code from JSP pages we are able to use Standard actions, but which are available in very less number and having bounded functionalities, which are not sufficient to remove java code completely from jsp pages.  
To remove java code completely from jsp pages.
- To remove java code completely from jsp pages, we are able to use Custom Actions along with Standard Actions, but in case of Custom tags, to implement simple java syntaxes like if, switch, for loop, ...
- We have to provide lot of java code in the form of tag handler classes.
- To overcome the above problems and to remove java code completely from jsp pages we have to use JSTL tags along with Standard actions and Custom Actions.
- JSTL has provided predefined tags to represent all the frequently used java syntaxes and actions in jsp pages in order to remove java code from JSP pages.
- JSTL is an abstraction provided by sun microsystem and its implementations are provided by all the server vendors.

→ Tomcat has provided JSTL implementation in the form of following JAR files in Tomcat Server at the location ↳

"C:\Tomcat 8.0\Debapps\examples\WEB-INF\lib"

- Standard.jar
- JSTL.jar

→ There are five types of tags in JSTL.

- i) Core Tags → <http://java.sun.com/jstl/core>
- ii) XML Tags → <http://java.sun.com/jstl/XML>
- iii) Formatted Tags → <http://java.sun.com/jstl/fmt>
- iv) SQL Tags → <http://java.sun.com/jstl/SQL>
- v) Functions Tags → <http://java.sun.com/jsp/jstl/functions>

### Core Tags:-

→ The main purpose of core tag library is to implement frequently used java syntaxes and some URL based actions.

There are four types of core tags.

#### i) General Purpose Tags:-

`<C:set>`, `<C:out>`, `<C:remove>`, `<C:catch>`

#### ii) Conditional Tags:-

`<C:if>`, `<C:choose>`, `<C:when>`, `<C:otherwise>`

#### iii) Iterative Tags:-

`<C:foreach>`, `<C:forTokens>`

#### iv) URL-Based Tags:-

`<C:import>`, `<C:curl>`, `<C:redirect>`

≡

### <c:set>

→ This tag can be used to set a key-value pair in a particular JSP Scope.

Syntax:-

<c:set var="---" value="---" scope="---"/>

→ Where "Var" and "value" attributes will take key-value data, where scope attribute will take a particular JSP scope to put key-value pair.

### <c:out>

→ This tag can be used to display data on client browser it's equal to out.println() method.

Syntax:-

<c:out value="---"/>

→ Where "value" attribute will take data which we want to display on client browser.

→ If we want to display a particular variable value available in either of the JSP scopes then we have to use an expression as value to "value" attribute that is "\$ {var-Name}".

### <c:remove>

→ This tag can be used to remove a key-value pair from JSP scopes.

Syntax:-

<c:remove var="---" scope="---"/>

→ Where "var" attribute will take "key" to remove key-value pair. Where "Scope" attribute will take a particular JSP scope where the required key-value pair is existed.

≡

## <c:catch>

→ This tag can be used to catch an ~~exception~~ which is generated in the body of <c:catch> tag.

Syntax:-

<c:catch var="---">

— java code to rise an exception —

</c:catch>

→ where "var" attribute will take a variable to store the generated exception object reference.

jstlapp

| → jstl.jsp

| → WEB-INF

    | → lib

        | → jstl.jar

        | → Standard.jar

| → Classes.

Ex:-

```
<%@taglib uri="http://java.sun.com/jstl/core" prefix="c"%>
```

```
<%@page isELIgnored="true"%>
```

```
<c:set var="a" value="AAA" scope="session"/>
```

```
<h1>
```

```
<%
```

```
    out.println("a → " + session.getAttribute("a"));
```

```
<%>
```

```
<br>
```

```
<c:out value="Amarendra Singh"/><br>
```

```
    a → <c:out value="${a}" /><br>
```

```
    <c:remove var="a"/>
```

```
    a → <c:out value="${a}" />
```

```
</h1>
```

`<C:catch var="e">`

`<Y.`

`<?+ a=10;`

`<?+ b=0;`

`float f=a/b;`

`%>`

`<IC:catch>`

`<h1><C:out value="$[e]" /></h1>`

`<C:if>`

→ This tag is representing "if" conditional statements, it will execute body when the specified condition is true.

Syntax:-

`<C:if test="-->`

                body —

`</C:if>`

→ Where "test" attribute will take conditional expression.

`<C:choose>`, `<C:when>`, `<C:otherwise>`

→ These tags are representing "switch" conditional statement.

Syntax:-

`<C:choose>` → Switch

`<C:when test="-->` Body `</C:when>` → Case

`<C:otherwise>` Body `</C:otherwise>` → default case.

`</C:choose>`

→ Where "test" attribute will take a Conditional expression.

Z

Ex:-

```
<c:set var="a" value="10"/>
<c:if test="#{a==10}>
<h1>a value is 10</h1>
</c:if>
<h1>
<c:choose>
  <c:when test="#{a==5}>a value is 5</c:when>
  <c:when test="#{a==10}>a value is 10</c:when>
  <c:when test="#{a==15}>a value is 15</c:when>
  <c:otherwise>No, Not in 5,10 and 15</c:otherwise>
</c:choose>
</h1>
```

### (c:forEach)

- This tag is implementing both for loop and foreach loop.
- If we want to use this tag as 'for' loop to perform iterations over body then we have to use the following.

Syntax:-

```
<c:forEach var="---" begin="---" end="---" step="---">
  — Body —
</c:forEach>
```

- Where "var" attribute will take a variable to store index value at each and every iteration.
  - Where "begin" attribute will take start index value.
  - Where "end" attribute will take end index value.
  - Where "step" attribute will take increment length.
-

→ If we want to use this tag foreach loop then we have to use the following syntax.

```
<c:foreach var="--" items="-->  
  — Body —  
</c:foreach>
```

→ Where "var" attribute will take a variable to store array element at each and every iteration.

→ Where "items" attribute will take array reference variable which is available in either of the JSP scopes.

### <c:forTokens>

→ This tag will perform String Tokenization on the provided String data.

#### Syntax:-

```
<c:forTokens var="--" items="--" separator="-->
```

— Body —

```
</c:forTokens>
```

→ Where "var" attribute will take a variable to store the generated token at each and every iteration.

→ Where "items" attribute will take string data to perform tokenization.

→ Where "separator" attribute will take a separator to perform tokenization.

#### Ex:-

```
<c:foreach var="a" begin="0" end="10" step="2">
```

```
<h1><c:out value="${a}"/></h1>
```

```
</c:foreach>
```

```
<%
```

```
String[] str={"AAA", "BBB", "CCC", "DDD"};
```

```
request.setAttribute("str", str);
```

```
%>
```

```

<c:foreach var="\$" items="\${str1}">
  <h1><c:out value="\${\$}"></h1>
</c:foreach>
<c:set var="data" value="Amarendra Singh"/>
<c:forTokens var="tokens" items="\${data}" * = " " >
  <h1><c:out value="\${tokens}"></h1>
</c:forTokens>

```

### <c:import>

→ This tag can be used to get the content of the specified target page into present jsp page.

#### Syntax:-

```
<c:import url="---"/>
```

→ Where url attribute will take name and location of the target page.

### Ex:- jello.jsp

```

<%@ taglib uri="http://java.sun.com/core" prefix="c"%>
<%@ page isELIgnored="true"%>
<h1>header</h1>
<c:import url="body.jsp"/>
<h1>Footer</h1>

```

### body.jsp

```
<h1>
```

Body

```
</h1>
```

z

### (C:url)

→ This tag can be used to represent the specified url.

#### Syntax:-

`(C:url value = "---")`

→ Where "Value" attribute will take an url to represent.

Ex:- `<a href = "(C:url value = "http://localhost:1010/forwardapp"/)">`  
`Home</a>`

### (C:redirect)

→ This tag is able to implement send Redirect mechanism in web applications.

#### Syntax:-

`(C:redirect url = "---")`

→ Where url attribute will take an url to which we want to redirect request.

#### Ex:-

`(C:redirect url = "http://localhost:1010/forwardapp"/).`

2

### Formatted Tags:-

→ The main purpose of formatted tags is to implement Internationalization [I18N] of jsp pages.

→ To implement Internationalization in jsp pages, JSTL has provided the following tags.

#### ① (fmt:setLocale)

→ This tag will set a particular locale to the current JSP page if we haven't used this tag then container will take browser provided locale.

`(fmt:setLocale value = "---")`

→ Where "Value" attribute will take local like "en-US", "it-IT".

2

## ② (fmt:formatNumber)

→ This tag will format the provided number w.r.t the specified locale.

`<fmt:formatNumber var="---" value="---"/>`

→ Where "var" attribute will take a variable to store the converted number.

→ Where "value" will take a number to convert.

## ③ (fmt:formatDate)

→ This tag will format current system date w.r.t a particular locale.

`<fmt:formatDate var="---" value="---"/>`

→ Where "var" attribute will take a variable to store converted data. Where "value" attribute will take Date object reference variable, it must be created by using (jsp:useBean) tag.

## ④ (fmt:setBundle)

→ This tag will create resource Bundle object on the basis of the specified properties files base name.

Syntax:-

`<fmt:setBundle basename=" " />`

→ Where basename attribute will take properties file base name.

## ⑤ (fmt:message)

→ This tag will get message on the basis of the specified key.

`<fmt:message var="---" key="---"/>`

→ Where "var" attribute will take a variable to store the message. Where "key" attribute will take message key which we specified in the properties file.

Note:- We have two properties files either under "classes" folder or under "src" folder.

## abc\_en\_US properties.

Welcome = Welcome to en US user.

## abc\_it\_IT properties.

Welcome = Welcome to it IT user.

```
Ex:- <%@taglib uri="http://java.sun.com/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt"%>
<%@page isELIgnored="true"%>
<fmt:setLocale value="en_US"/>
<fmt:formatNumber var="num" value="123456.23456789"/>
<h1><c:out value="${num}"/></h1>
<jsp:useBean id="d" class="java.util.Date"/>
<fmt:formatDate var="date1" value="${d}" type="both"/>
<fmt:formatDate var="date2" value="${d}" type="date"
dateStyle="LONG"/>
<fmt:formatDate var="date3" value="${d}" type="time"/>
<jsp:useBean>
<h1>
<c:out var="${date1}"/>
<c:out value="${date2}"/>
<c:out value="${date3}"/>
</h1>
<fmt:bundle basename="abc">
<fmt:message var="msg" key="welcome"/>
<h1><c:out value="${msg}"/></h1>.
```

R

## SQL Tags:-

- These tags can be used to interact with database in order to perform database operations from jsp pages.
- To perform database operations from JSP pages, JSTL has provided the following tags.

### ① <sql:setDataSource>

- This tag can be used to setup the JDBC environment like loading the Driver and establish the connection with database.  
`<sql:setDataSource driver="---" url="---" user="---" password="---"/>`
- Where "driver", "user", "url" and password attributes can be used to provide all the jdbc parameters like driver class name, driver url, database username and database password.

### ② <sql:update>

- This tag can be used to execute non-select sql queries like Create, insert, update, delete, ...

Syntax:-

```
<sql:update var="---" sql="sql query"/>
```

Syntax(ex)

```
<sql:update var="---">
```

— SQL query —

```
(/sql:update)
```

- If we provide sql queries in Prepared Statement style then we have to use `<sql:param>` tags to set values to the positional parameters.

```
<sql:param value="---"/> or
```

```
<sql:param> value</sql:param>
```

page 70 - 71, 72, 73, 74  
Application → 23, 24, 25, 26, 27

### {S91:query}

→ This tag can be used to execute source S91 query in order to retrieve data from database table.

{S91:query var = "\_\_\_" S91 = "\_\_\_"}

{S91:query var = "\_\_\_"}

S91 query .

{/S91:query}

### {S91:transaction}

→ It will represent a single transaction in TSP pages.

{S91:transaction}

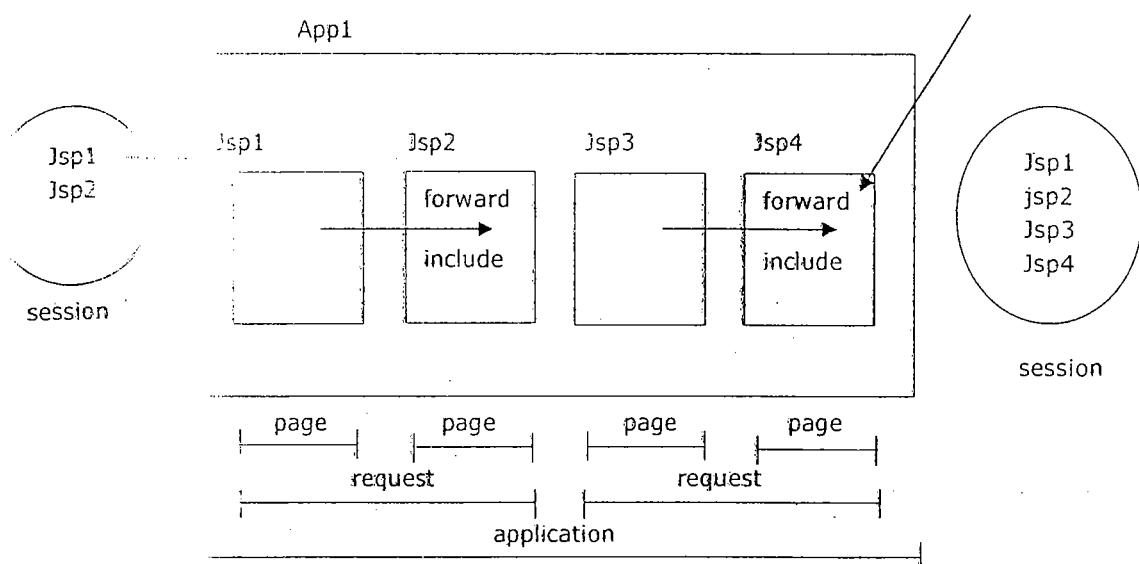
— — — —

{/S91:transaction}

≡

page 70:- 74  
Application : - 28

≡



## 1. Page Scope:

If we declare the data in page scope by using pageContext object then that data should have the scope up to the present Jsp page.

## 2. Request Scope:

If we declare the number of resources which are visited by the present request object.

## 3. Session Scope:

If we declare the data in HttpSession object then that data should have the scope up to the number of resources which are visited by the present client.

## 4. Application Scope:

If we declare the data in ServletContext object then that data should have the scope up to the number of resources which are available in the present web application.

### Application1-----

#### App1:

Employees.jsp

s.html

```
<html>
    <body bgcolor="lightblue"><br><br><br><br>
    <center><h1>Employee Details Form</h1></center>
    <pre><h2>
<form method="get" action="display.jsp">
    Employee Id :<input type="text" name="eid"/>
    Employee Name : <input type="text" name="ename"/>
    Employee Salary : <input type="text" name="esal"/>
    <input type="submit" value="Display"/>
</h2></pre>
</body>
</html>
```

### display.jsp:

```
<%!
    int eid;
    String ename;
    float esal;
%>
<%
try {
    eid=Integer.parseInt(request.getParameter("eid"));
    String ename=request.getParameter("ename");
    float esal=Float.parseFloat(request.getParameter("esal"));
}
catch(Exception e){
    e.printStackTrace();
}
%>
<html>
    <body>
        <center><h1>Employee Details</h1></center>
        <center>
            Employee Id : <%=eid %><br><br>
            Employee Name : <%=ename %><br><br>
            Employee Salary : <%=esal %><br><br>
        </center>
    </body>
</html>
```

## 3. Jsp Actions:

In Jsp technology, by using scripting elements we are able to provide java code inside the Jsp pages, but the main theme of Jsp technology is not to allow java code inside Jsp pages.

To eliminate java code from Jsp pages we have to eliminate scripting elements, to eliminate scripting elements from Jsp pages we have to provide an alternative i.e. Jsp Actions.

**usebeanapp:****empform.html:**

```
<html>
    <body bgcolor="lightblue"><br><br><br><br>
    <center><h1>Employee Details Form</h1></center>
    <form method="get" action="display.jsp">
        <pre><h2>
            Employee Id : <input type="text" name="eid"/>
            Employee Name : <input type="text" name="ename"/>
            Employee Salary : <input type="text" name="esal"/>
            <input type="submit" value="Display"/>
        </h2></pre>
    </form>
    </body>
</html>
```

**Employee.java:**

```
package comm.dss;
public class Employee implements java.io.Serializable {
    private int eid;
    private String ename;
    private float esal;
    public int getEid() {
        return eid;
    }
    public void setEid(int eid) {
        this.eid = eid;
    }
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public float getEsal() {
        return esal;
    }
    public void setEsal(float esal) {
        this.esal = esal;
    }
}
```

**display.jsp:**

```
<%!
    int eid;
    String ename;
    float esal;
%>
```

```

<%
try {
    eid=Integer.parseInt(request.getParameter("eid"));
    ename=request.getParameter("ename");
    esal=Float.parseFloat(request.getParameter("esal"));
}
catch(Exception e){
    e.printStackTrace();
}

%>
<jsp:useBean id="e" class="com.dss.EmployeeBean" type="com.dss.EmployeeBean"
scope="session">
<jsp:setProperty name="e" property="eid" value='<%=eid %>'/>
<jsp:setProperty name="e" property="ename" value='<%=ename %>'/>
<jsp:setProperty name="e" property="esal" value='<%=esal %>'/>
<html>
    <body>
        <center><h1>Employee Details</h1></center>
        <center>
            Employee Id : <jsp:getProperty name="e"
property="eid"/><br><br>
            Employee Name : <jsp:getProperty name="e"
property="ename"/><br><br>
            Employee Salary : <jsp:getProperty name="e"
property="esal"/><br><br>
        </center>
    </body>
</html>
</jsp:useBean>

```

**Note:** In case of `<jsp:useBean>` tag, in general we will provide a separate `<jsp:setProperty>` tag to set a particular value to the respective property in Bean object.

In case of `<jsp:useBean>` tag, it is possible to copy all the request parameter values directly onto the respective Bean object.

To achieve this we have to provide "\*" as value to property attribute in `<jsp:setProperty>` tag.

**Ex:** `<jsp:setProperty name="e" property="*"/>`

If we want to achieve the above requirement then we have to maintain same names in the request parameters i.e. form properties and Bean properties.

**Note:** The above "\*" notation is not possible with `<jsp:getProperty>` tag.

#### 4. `<jsp:include>`:

**Q:** What are the differences between include directive and `<jsp:include>` action tag?

```

public int doStartTag()throws JspException {
    return SKIP_BODY;
}
public int doAfterBody()throws JspException {
    return SKIP_BODY;
}
public int doEndTag()throws JspException {
    return EVAL_PAGE;
}
public void release() { }
}

public class MyHandler implements TagSupport
{ ---- }

```

**-----Application7-----**

**custapp2:**

**iterate.jsp:**

```

<%@taglib uri="/WEB-INF/iterate.tld" prefix="mytags"%>
<mytags:iterate times="10"><br>
    Durga Software Solutions
</mytags:iterate>

```

**iterate.tld:**

```

<taglib>
    <jsp-version>2.1</jsp-version>
    <tlib-version>1.0</tlib-version>
    <tag>
        <name>iterate</name>
        <tag-class>com.dss.Iterate</tag-class>
        <body-content>jsp</body-content>
        <attribute>
            <name>times</name>
            <required>true</required>
            <rtpvalue>true</rtpvalue>
        </attribute>
    </tag>
</taglib>

```

**Iteration.java:**

```

package com.dss;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
public class Iterate extends TagSupport
{
    int count=1;
    private int times;
    public void setTimes(int times)
    {

```

```
        this.times=times;
    }
    public int doStartTag() throws JspException
    {
        return EVAL_BODY_INCLUDE;
    }
    public int doAfterBody() throws JspException
    {
        if(count<times)
        {
            count++;
            return EVAL_BODY_AGAIN;
        }
        else
            return SKIP_BODY;
    }
}
```

## Nested Tags:

Defining a tag inside a tag is called as **Nested Tag**.

In custom tags application, if we declare any nested tag then we have to provide a separate configuration in tld file and we have to prepare a separate TagHandler class under classes folder.

### -----Application8-----

#### **custapp3:**

#### **nested.jsp:**

```
<%@taglib uri="/WEB-INF/nested.tld" prefix="mytags"%>
<h1><center>
<mytags:if condition='<%=10>20%'>
    <mytags:true>condition is true</mytags:true>
    <mytags:false>condition is false</mytags:false>
</mytags:if>
</center></h1>
```

#### **nested.tld:**

```
<taglib>
    <jsp-version>2.1</jsp-version>
    <tlib-version>1.0</tlib-version>
```



```
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
public class True extends TagSupport
{
    public int doStartTag() throws JspException
    {
        If i=(If)getParent();
        boolean condition=i.getCondition();
        if(condition == true)
            return EVAL_BODY_INCLUDE;
        else
            return SKIP_BODY;
    }
}
```

**False.java:**

```
package com.dss;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
public class False extends TagSupport
{
    public int doStartTag() throws JspException
    {
        If i=(If)getParent();
        boolean condition=i.getCondition();
        if(condition == true)
            return SKIP_BODY;
        else
            return EVAL_BODY_INCLUDE;
    }
}
```

**-----Application9-----****custapp4:****empdetails.jsp:**

```
<%@taglib uri="/WEB-INF/emp.tld" prefix="emp"%>
<emp:empDetails/>
```

**emp.tld:**

```
<taglib>
    <jsp-version>2.1</jsp-version>
    <tlib-version>1.0</tlib-version>
    <tag>
        <name>empDetails</name>
        <tag-class>com.dss.EmpDetails</tag-class>
```

```
<body-content>empty</body-content>
</tag>
</taglib>
```

**EmpDetails.java:**

```
package com.dss;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.sql.*;
public class EmpDetails extends TagSupport
{
    Connection con;
    Statement st;
    ResultSet rs;
    public EmpDetails()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system",
durga");
            st=con.createStatement();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    public int doStartTag() throws JspException
    {
        try
        {
            JspWriter out=pageContext.getOut();
            rs=st.executeQuery("select * from emp");
            ResultSetMetaData rsmd=rs.getMetaData();
            int count=rsmd.getColumnCount();
            out.println("<html><body bgcolor='pink'>");
            out.println("<center><br><br>");
            out.println("<table border='1' bgcolor='lightyellow'>");
            out.println("<tr>");
            for (int i=1;i<=count;i++)
            {
                out.println("<td><b><font size='6'>
color='red'><center>"+rsmd.getColumnName(i)+"</center></font></b></td>\"");
            }
            out.println("</tr>");
            while (rs.next())
            {
                out.println("<tr>");
                for (int i=1;i<=count;i++)
                {
```

```

        out.println("<td><b><font
size='5'>" + rs.getString(i) + "</font></b></td>");
    }
    out.println("</tr>");
}
out.println("</table></center></body></html>");
}
catch (Exception e)
{
    e.printStackTrace();
}
return SKIP_BODY;
}
}

```

### 3. Body Tags:

Up to now in our custom tags (simple classic tags, iteration tags) we prepared custom tags with the body, where we did not perform updatations over the custom tag body, just we scanned custom tag body and displayed on the client browser.

If we want to perform updatations over the custom tag body then we have to use Body Tags.

If we want to design body tags in Jsp technology then the respective TagHandler class must implement BodyTag interface either directly or indirectly.

```

public interface BodyTag extends IterationTag
{
    public static final int EVAL_BODY_INCLUDE;
    public static final int SKIP_BODY;
    public static final int EVAL_PAGE;
    public static final int SKIP_PAGE;
    public static final int EVAL_BODY_AGAIN;
    public static final int EVAL_BODY_BUFFERED;
    public void setPageContext(PageContext pageContext);
    public void setParent(Tag t);
    public Tag getParent();
    public int doStartTag() throws JspException;
    public void doInitBody() throws JspException;
    public void setBodyContent(BodyContent bodyContent);
    public int doAfterBody() throws JspException;
    public int doEndTag() throws JspException;
    public void release();
}

public class MyHandler implements BodyTag
{ ---- }

```

In case of body tags, there are 3 possible return values from doStartTag() method.

1. EVAL\_BODY\_INCLUDE

```

<%@taglib uri="http://java.sun.com/jstl/sql" prefix="sql"%>
<%@page isELIgnored="true"%>
<html>
    <body>
        <center><b><font size="7">
            <sql:setDataSource driver="oracle.jdbc.driver.OracleDriver"
url="jdbc:oracle:thin:@localhost:1521:xe" user="system" password="durga"/>
            <sql:update var="result">
                delete emp where esal>1000
            </sql:update>
            Row Count ... <c:out value="${result}" />
        </font></b></center>
    </body>
</html>

```

### 3. <sql:query---->:

This tag can be used to execute selection group SQL queries in order to fetch the data from database table.

**Syntax 1:** <sql:query var="--" sql="--"/>

**Syntax 2:** <sql:query var="--" ----- query ----- </sql:query>

If we execute selection group SQL queries by using <sql:query> tag then SQL tag library will prepare result object to hold up fetched data.

In SQL tag library, result object is a combination of ResultSet object and ResultSetMetaData object.

In result object, all the column names will be represented in the form a single dimensional array referred by columnNames predefined variable and column data (table body) will be represented in the form of 2-dimensionnal array referred by rowsByIndex predefined variable.

### -----Application28-----

#### sql5.jsp:

```

<%@taglib uri="http://java.sun.com/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jstl/sql" prefix="sql"%>
<%@page isELIgnored="true"%>
<html>
    <body>
        <center><b><font size="7">
            <sql:setDataSource driver="oracle.jdbc.driver.OracleDriver"
url="jdbc:oracle:thin:@localhost:1521:xe" user="system" password="durga"/>
            <sql:query var="result" sql="select * from emp"/>
            <table border="1" bgcolor="lightyellow">
                <tr>

```

```

<c:forEach var="columnName"
items="${result.columnNames}">
    <td><center><b><font size="6" color="red">
        <c:out value="${columnName}"/>
    </font></b></center></td>
</c:forEach>
</tr>
<c:forEach var="row" items="${result.rowsByIndex}">
    <tr>
        <c:forEach var="column" items="${row}">
            <td><b><font size="5">
                <c:out value="${column}"/>
            </font></b></td>
        </c:forEach>
    </tr>
</c:forEach>
</table>
</font></b></center>
</body>
</html>

```

#### 4. <sql:transaction---->:

This tag will represent a transaction, which includes collection of <sql:update> tags and <sql:query> tags.

## 3. I18N Tags(F\$formatted Tags):

### 1. <fmt:setLocale---->:

This tag can be used to represent a particular Locale.

**Syntax:** <fmt:setLocale value="--"/>

Where value attribute will take Locale parameters like en\_US, it\_IT and so on.

### 2. <fmt:formatNumber---->:

This tag can be used to represent a number w.r.t the specified Locale.

**Syntax:** <fmt:formatNumber var="--" value="--"/>

Where var attribute will take a variable to hold up the formatted number.

Where value attribute will take a number.

### 3. <fmt:formatDate---->:

This tag can be used to format present system date w.r.t. a particular Locale.

**Syntax:** <fmt:formatDate var="--" value="--"/>