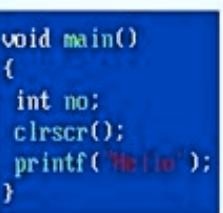




  
**College  
Projects**

  
**Basic  
Program**

**C  
Tutorial**

  
**JAVA  
Programming**



**Tutorial4Us**

Tutorials for Beginners



**Tutorial4Us**

Tutorials for Beginners



**Tutorial4Us**

Tutorials for Beginners

# **tutorial4us.com**

**A Perfect Place for All Tutorials Resources**

Core Java | Servlet | JSP | JDBC | Struts | Hibernate | Spring

Java Projects | C | C++ | DS | Interview Questions | JavaScript

College Projects | eBooks | Interview Tips | Forums | Java Discussions

**For More Tutorials Stuff Visit**

# **www.tutorial4us.com**

**A Perfect Place for All Tutorials Resources**

# Advance Java

(Natrajz Sir Notes)

[www.tutorial4us.com](http://www.tutorial4us.com)

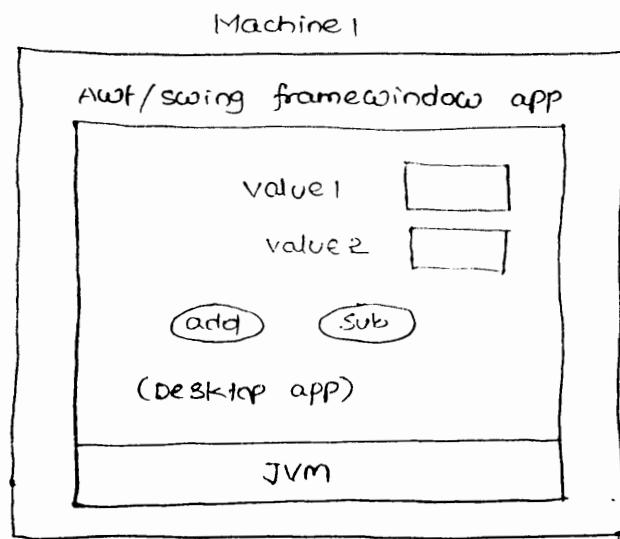
# Advanced Java

- \* Since java can be used to develop all kinds of software applications so it is called as programming suite.
- \* Java can be divided into 3 modules. They are
  - (i) JSE - Java Standard Edition
  - (ii) JEE - Java Enterprise Edition
  - (iii) JME - Java Mobile/Micro Edition

## JSE:

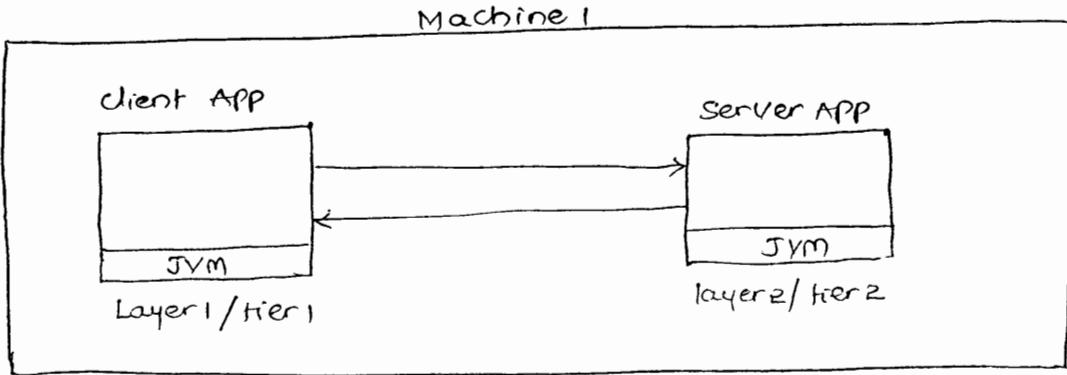
- \* It is installable software as jdk software.
- \* Latest version is 7.0 (Dolphin)
- \* Java 6.0 is called as mustang and Java 5.0 is called as tiger.
- \* This module is given to develop stand-alone applications, desktop applications, two-tier applications . . .
- \* The application that is specific to one computer and contains main() is called as standard or stand-alone application.
- \* The stand-alone application that contains GUIness is called as desktop application.

Ex: Awt framewindow application, swing framewindow application.

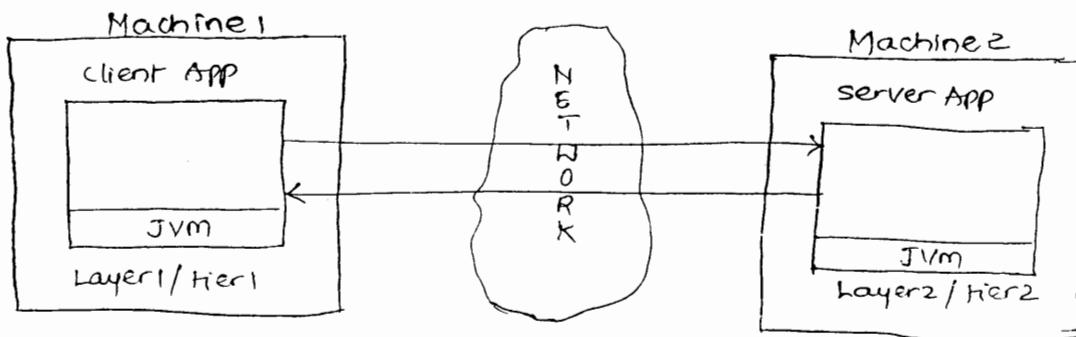


- \* The application that contains two layers communicating with each other is called as two-tier application.
- \* ↳ layer represents logical partition in the application having logics.

- \* The layers of two-tier application can be there in a single computer (or) can be there in two different computers.



socket programming based client-server application (two-tier app)



socket Programming based client-server application (two-tier app)

- \* To execute multiple Java applications parallelly or simultaneously from a single computer, multiple JVMs will be activated simultaneously (or) parallelly.
- \* Applet is not desktop application. It is a compiled Java class that can be sent over the network as webpage giving alternate to the HTML based form pages.

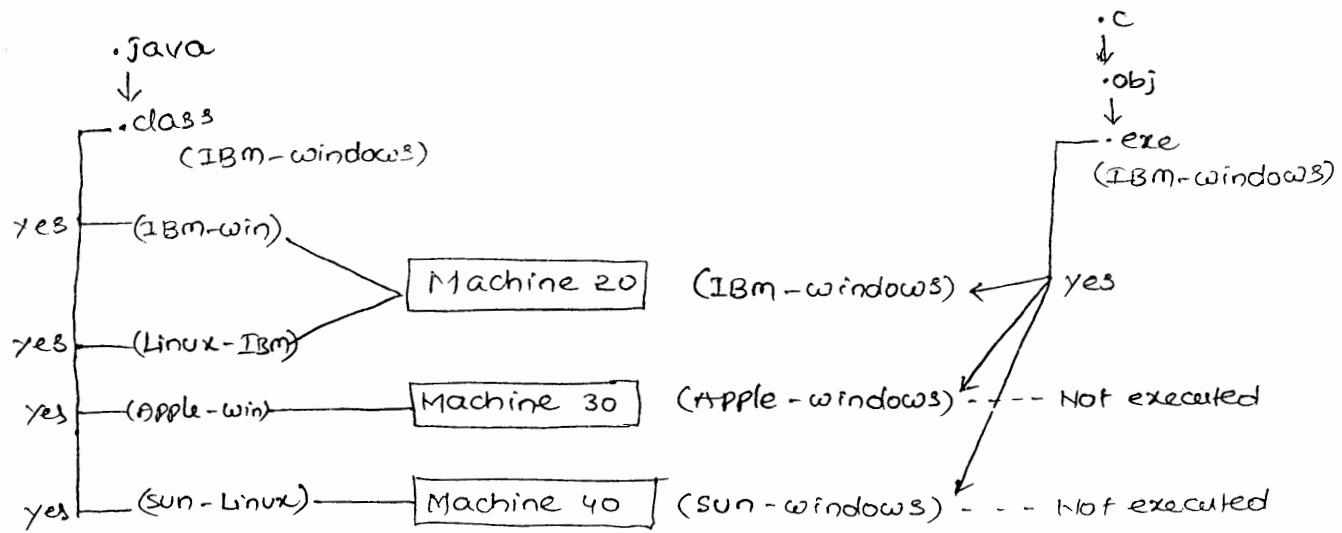
JSE Concepts: awt, swing, collection frame work, JDBC, JNDI (Java Naming and Directory Interface) etc.

\* The alternate technologies for JSE module are VB, VB.NET, D2K. But these technologies based applications are platform dependent and architectural specific whereas the JSE module applications are architectural neutral and platform independent.

As of now three popular architectures are there to manufacture computers. Each architecture use methodology and plan of manufacturing

computer. They are

- (1) IBM architecture
- (2) Apple architecture
- (3) SUN architecture.



\* The c, c++ generated .exe files not only contains operating system related instructions but they also contain instructions related to underlying computer architecture. Due to this we can say c, c++ languages are platform dependent and architecture specific whereas java programming language is platform independent and architectural neutral because the java compiler generated .class file doesn't contain underlying OS, computer architecture instructions.

#### JEE:

\* JEE means Java Enterprise Edition.

\* It is not installable software, it is given as specification.

\* JEE specification contains rules and guidelines to develop webserver and application server software like weblogic, tomcat etc.

NOTE: Working with JEE is nothing but working with one or other web-server or app server software to develop the app.

\* These webserver, application server softwares are installable softwares.

\* For JEE module JSE module is base module.

\* Using JEE module the following applications can be developed.

- (1) Web applications (websites)
- (2) Distributed applications
- (3) Enterprise applications
- (4) n-tier applications

\* The applications which can provide 24x7 access to their resources using internet environment are called as websites.

\* The client-server applications which provide location transparency are called as distributed applications. If client application is able to recognize the server application location change dynamically are called as location transparency.

\* A web application / website can be developed as distributed or non-distributed application.

\* All credit card, debit card processing applications will be developed generally as distributed applications.

\* A large scale application that deals with complex and heavy weight business logic by having middleware service support is called as enterprise application.

Ex: Banking application.

\* The additional services that are configurable on the applications to make applications running perfectly in all the situations are called as middleware services.

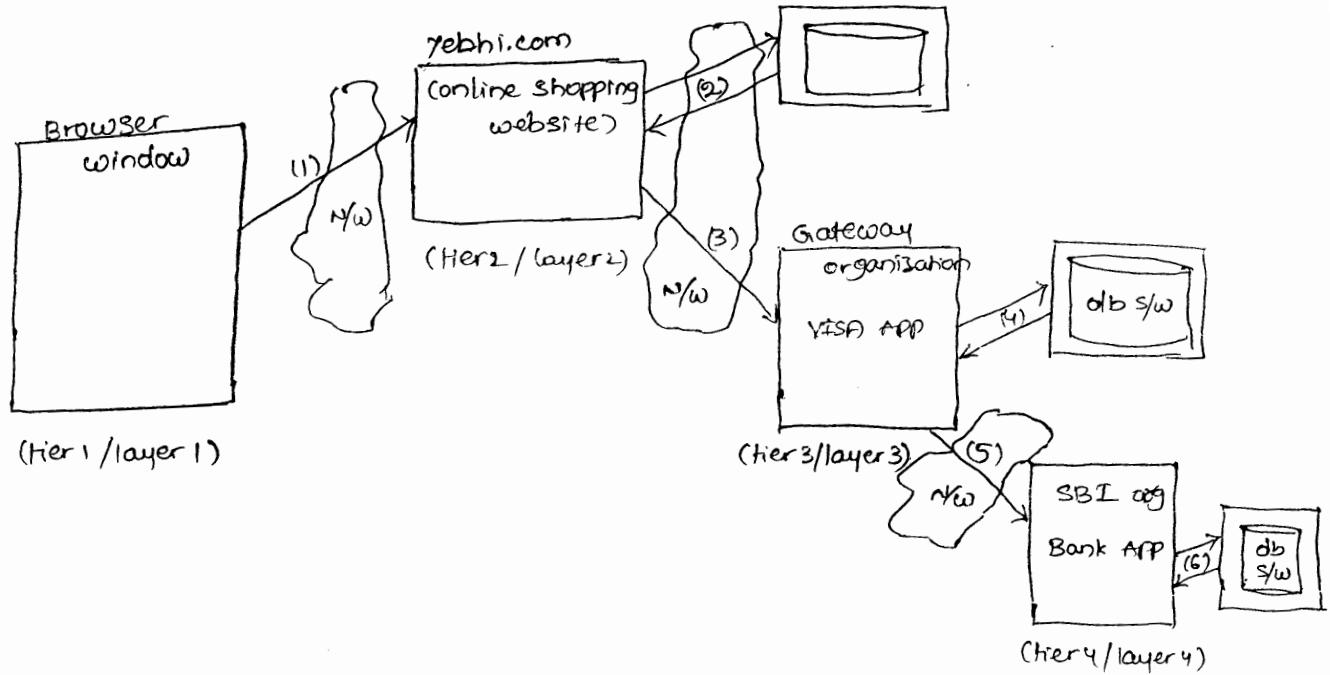
Ex: security service, transaction management service, pooling service and etc..

NOTE: Middleware services are not minimum logics of the application, they are additional and optional logics to develop.

\* The application that contains multiple layers/tiers (more than two) is called as n-tier applications.

Ex: online shopping website environment.

## online shopping website environment (n-tier APP)



- \* Generally enterprise applications will be developed as n-tier applications.
- \* JEE module concepts are servlets, JSP, EJB, JTA (Java Transaction API), JMS (Java messaging service), Java mail, JAAS (Java Authentication and Authorization service), JMX (Java management console), JCA (Java Connector API) and etc...
- \* Struts, spring and hibernate technologies are not part of Sunmicrosystems JEE module. They are given by third party organizations as alternate and enhancement to JEE concepts.
- \* While working with all JEE Concepts we must take support of webserver (or) application server softwares.
- \* majority projects of java environment will be developed based on JEE module.
- \* Every project belongs one domain. This JEE module is suitable for developing banking financial service, Insurance, health care domain projects.
- \* JEE module is suitable to develop gaming, automation and retail domain projects.

- \* JME module is suitable for developing mobile gaming, telecommunication domain projects.
- \* The latest version of JME module is 6.

### JME:

- \* JME means Java mobile/micro edition.
- \* For JME module JSE module is base module.
- \* It is installable software.
- \* Latest version is 2.0
- \* Given to develop mobile and micro applications in Java environment like sim cards, mobile games and etc.
- \* Use wml (wireless markup language), wap in the application development.
- \* These applications can be embedded with different types of devices (electronic, electrical, automobile and etc...).
- \* JSE, JEE module based applications can not be attached with (or) embedded with different types of devices by bringing the code outside the computer. But this is possible with JME module.
- \* The above process is called as embedded system programming and this is useful to add artificial intelligence to devices, that means we can make the devices to think and take the decisions.  
Ex: A fully automated washing machine, Robot.
- \* To develop mobile related and artificial intelligence related applications in java environment use JME module.

### API (Application programming Interface):

- \* It is the base for the programmer to develop certain software technology based software applications.
- \* Every software technology used built-in APIs. Using these APIs the programmers can develop user defined APIs and software applications.

\* In "C" language API is set of functions which come in the form of header files.

\* In "C++" language API is set of functions, classes which come in the form of header files.

\* In "Java" language API is set of classes, interfaces, enums, annotations which come in the form of packages.

### Examples of Java APIs :

(1) AWT API → working with java.awt, java.awt.event packages.

(2) Reflection API → working with java.lang.reflect package

(3) JDBC API → working with java.sql, javax.sql packages

(4) Utility API → working with java.util package

\* We can see three types of APIs in Java.

(1) Built-in APIs → given by Sun micro system along with Jdk softwares.

Ex: AWT, JDBC etc...

(2) User defined APIs → given by programmers

(3) Third party APIs → given by third party organizations (other than sun micro systems and programmers).

Ex: Java Zoom API (given by JavaZoom organization, downloadable from www.javazoom.net).

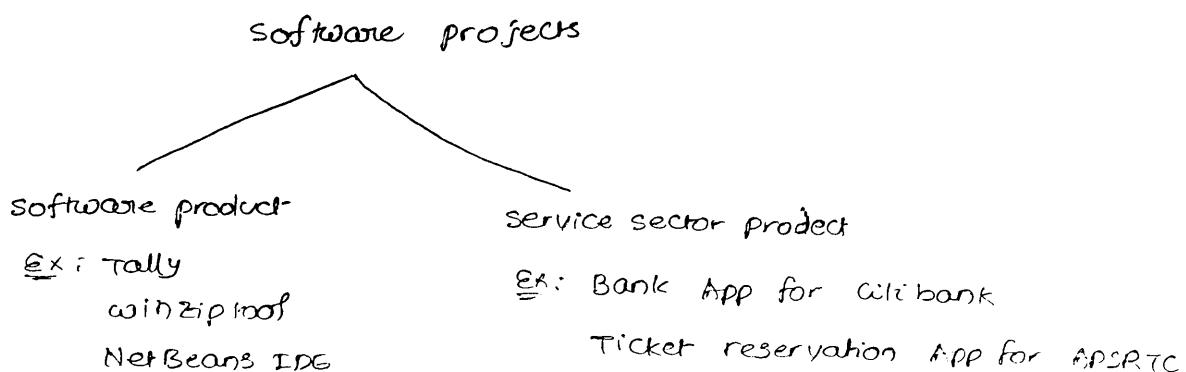
\* Generally the Java related APIs come as in the form of jar files. (Java style zip file).

\* JAVA-HOME\jre\lib\rt.jar file represents all the built-in APIs of Jdk software.

\* The company who creates softwares is called as software vendor company.

Ex: IBM, Sun micro system, Oracle corporation and etc.

- \* The organization who gives their requirements to software development company is called as client organization.  
Ex: City bank, SBI, APSRTC etc..
- \* The company who takes the support of softwares given by software vendor companies and develop the project for client organizations is called as software development company.  
Ex: Infosys, Wipro, Polaris etc..



- \* If project is designed and developed exclusively for one client organization by gathering requirements, then it is called as service sector project.
- \* If project is developed based on common requirements and given to multiple clients then it is called as software product.
- \* For service sector project client organization will be there before development of the project.
- \* For software products client organizations will be there after the development of product.

What is the difference between static block and constructor?

- A) Static block is class level one time execution block. When JVM loads java class the static block of that class executes automatically. This block is useful to initialize static member variables of a class.

constructor is object level one time execution block

when JVM creates object for a class then the constructor of that class executes once automatically. Constructor is useful to initialize instance variables of ~~java~~ class object.

\* `Class.forName()` makes JVM of the current application to load given class into application from the memory. But this method does not create object of the loader class.

example application to understand above concepts.

```
class Demo
{
    static
    {
        System.out.println ("Demo: static block");
    }

    Demo()
    {
        System.out.println ("Demo: Constructor");
    }
}

class Test
{
    static
    {
        System.out.println ("Test: static block");
    }

    Test()
    {
        System.out.println ("Test: Constructor");
    }
}

public class TestAPP
{
    static
    {
        System.out.println ("TestAPP: static block");
    }

    public static void main (String args[])
    {
        System.out.println ("TestAPP: main()");
        Demo d1 = new Demo();
        Demo d2 = new Demo();
    }
}
```

```
    class.forName("Test");
    class.forName("Test");
```

↓      ↓

O/P    TestApp: static block

```
TestApp: main()
Demo: static block
Demo: Constructor
Demo: Constructor
Test: static block
```

\* with respect to above application we can see JVM loading java classes in the following situations.

- (1) while creating first object for given java class
- (2) when `class.forName()` method is called.
- (3) when java class is given to "java tool" for execution.

```
> java TestApp
```

#### Understanding `Class.forName()`:

Prototype:

```
public static Class forName(String name) throws ClassNotFoundException
```

\* This method makes JVM to load given class or interface into application from memory dynamically at runtime. If they are not available to load this method throws checked exception called `ClassNotFoundException`.

\* It is a static method (can be called without object)

\* available in `java.lang.Class` (pre-defined)

\* It is factory method

\* A method of a java class that is capable of constructing and returning its own java class object is called as factory method.

Ex: Thread t = Thread.currentThread();

```
String s1 = String.valueOf(t);
```

```
Calendar c1 = Calendar.getInstance();
```

```
Class c = Class.forName("Test");
```

```
class c = Class.forName("Test");
```

- \* The above statement makes JVM to load class "Test" into application dynamically at runtime and returns object of (c) "java.lang.Class" referring the loaded class "test".
- \* In the above statement 'c' is not the object of the loaded class "Test", it is object of java.lang.Class referring the loaded class "Test". we can use this object 'c' to gather details about the loaded class "Test" to create object of loaded class "Test" and etc.. operations.
- \* If java method throws checked exception while calling that method we must handle the exception using try/catch blocks or we need to declare the exception to be thrown by using throws statement.

Example application based on all the above statements

```
// TestApp1.java
public class TestApp1
{
    public static void main (String args[])
        throws Exception
    {
        Class c = Class.forName (args[0]);
        //gathered details and loaded class and interface
        System.out.println(args[0] + " is an interface? " + c.isInterface ());
        System.out.println ("name of " + args[0] + " is: " + c.getName ());
        System.out.println ("super class of " + args[0] + " is: " + c.getSuperclass ());
    } //main
} // class
```

> javac TestApp1.java (Compile time)

> java TestApp1 java.lang.String (run time)

\* Command line argument args[0]

\* Inorder to know class name of any java object programmatically then call getClass () method on that object.

\* This method is available as public method of "java.lang.Object" class

Ex:    string s<sub>1</sub> = new String("ok");  
          System.out.println("class name of s<sub>1</sub> obj" + s<sub>1</sub>.getClass());  
          java.util.Date d<sub>1</sub> = new java.util.Date();  
          System.out.println("class name of d<sub>1</sub> obj" + d<sub>1</sub>.getClass());  
          Class c = Class.forName("java.awt.Button");  
          System.out.println("classname of obj 'c' is:" + c.getClass());

→ java.lang.String  
→ java.util.Date  
→ java.lang.Class

### Problem with "new" keyword:

\* "new" keyword creates the object of a Java class at runtime but it expects the presence of the class (or) availability of the class at compile time.

```
Test t = new Test();
```

\* In the above statement object 't' will be created at runtime but it expects the presence of class "Test" at compile time.

\* This indicates new keyword is not suitable to create the object of a class whose class name comes to application dynamically at runtime. To overcome this problem use "newInstance()" method of java.lang.Class.

\* The newInstance() method can create object of given Java class at runtime even though class name is supplied to the application dynamically at runtime. This kind of object creation process is required while performing drag and drop operations in GUI builder applications.

```

    catch (ClassNotFoundException cnf) // to handle known exception
    {
        cnf.printStackTrace();
    }
    catch (Exception e) // to handle all unknown exceptions.
    {
        e.printStackTrace();
    }
}
// main
} // class

> javac objTest.java (Compile time)
> java objTest Test (runtime)
> java objTest java.awt.Rectangle (runtime)

```

O/P Test: o - org constructor  
object data is: a=0 b=0.0

\* when the above newInstance() method is used to create object of abstract class or interface then that method raises "java.lang instantiation exception"

\* It is always recommended to catch and handle the exceptions by keeping multiple catch blocks after try block. This helps the programmer to handle both known and unknown exceptions and to differentiate the logics based on the exception that are raised (refer above).

what is the difference between <exception obj>.toString() and <exception obj>.printStackTrace() ?

\* to string() method prints name of the exception class along with description when exception is raised. whereas printStackTrace() prints the route cause and hierarchy in the application execution when exception is raised.

\* Every java method is executed in stack memory. This stack memory is filled up with messages when exception is raised during method execution.

\* The printStackTrace() call displays these messages on console monitor.

(Q) when java does not support pointers why does it throw null pointerException?

(A) When Java method is called on a variable that holds null value this nullPointerException will be raised. To solve this problem make that variable referring to object before calling the method.

Ex: Problem:

```
Date d;  
d.toString(); // raise java.lang.NullPointerException
```

solution :

```
Date d = new Date();  
d.toString(); // success
```

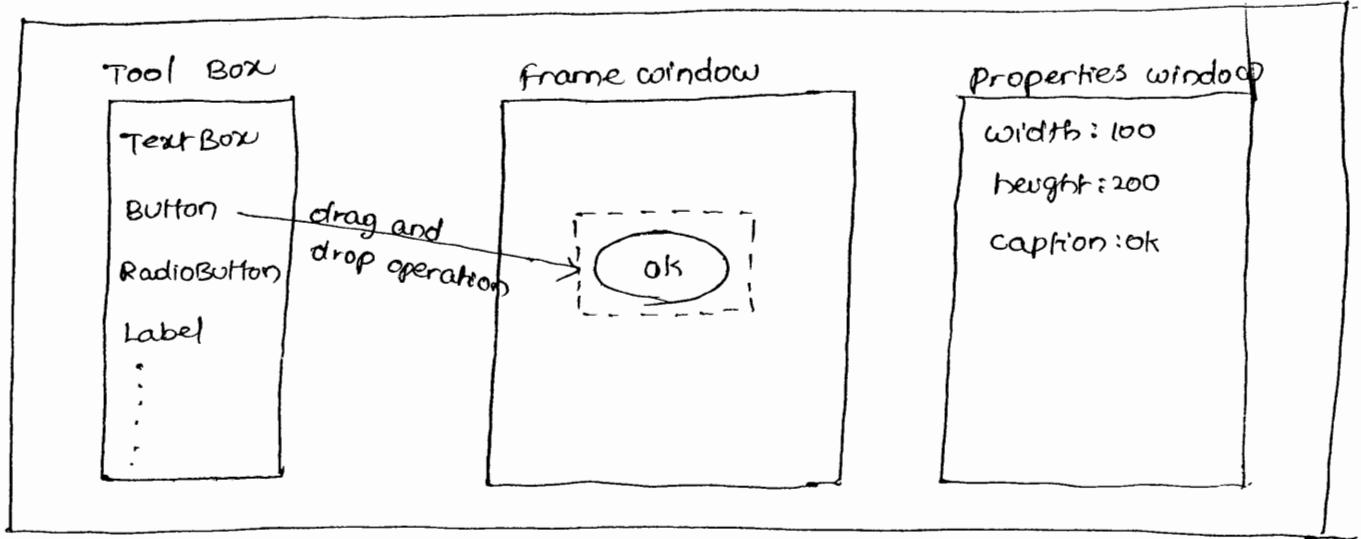
\* In java.lang.NullPointerException class name the word pointer is no way related with C, C++ pointers. Look at that word with dictionary meaning which says method is called on a variable that points to null value.

How many ways are there to create object of a java class?

There are multiple ways. They are

- (1) By using new keyword
- (2) By using static factory method
- (3) By using instance factory method
- (4) By using factory pattern
- (5) By using newInstance()
- (6) By using cloning process
- (7) By using deserialization process ...

## GUI Builder Application (Part of IDE SWs)



String s = new String ("oks");

s(String class obj)

ok  
(data of the obj)

class c = Class.forName("Test");

(The given class/interface to load)

c(obj of java.lang.class)

"Test"  
(data of obj)

\* In a running Java application if we want to refer a Java class or interface then we need to use the object of java.lang.Class

\* When you call Java method catching and using the return value of Java method is purely programmer choice.

Class.forName("Test"); // valid statement

Class c = Class.forName("Test"); // valid statement

+ When Java object is created constructor will execute and the instance variables will be assigned with default values.

creating object of java class by using newInstance() method  
by getting class name into application dynamically at runtime.

//Test.java

```
public class Test
{
    int a;
    float b;

    public Test()
    {
        System.out.println("Test: 0-arg constructor");
    }

    public String toString()
    {
        return "a=" + a + "b=" + b;
    }
}
```

>javac Test.java

//objTest.java

```
public class ObjTest
{
    public static void main (String args[])
    {
        try
        {
            // load given class into application
            Class c = Class.forName(args[0]);
            // "c" object refers to the loaded class
            // create object for loaded class
            Object obj = c.newInstance();
            // creates object of a java class
            // that is referred by object "c"
            System.out.println("object data is:" + obj.toString());
        }
        catch (InstantiationException iae) // to handle known exception
        {
            iae.printStackTrace();
        }
    }
}
```

### 1) By using new keyword

```
String s1 = new String("Hai");  
Date d = new Date();
```

### 2) By using static factory method

\* A method in a java class that is capable of constructing and returning its own java class object is called as static factory method.

```
Thread t = Thread.currentThread();  
String s1 = String.valueOf(t); } static factory methods  
class c = Class.forName("Test");
```

NOTE: when java class contains only private constructors to create object of that class outside the class we take the support of static factory methods.

Ex: java.lang.Class is having only private constructors so we always use Class.forName() method as shown above to create the object.

### 3) By using instance factory method:

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

```
String s1 = s.concat(" how are u"); // s1 holds "Hello how are u"  
↓  
instance  
factory method
```

\* Static factory methods will be called without object. Instance factory methods will be called by using object.

\* In order to create a new object for java class by using existing object and its data take the support of instance factory methods.

NOTE: In the above example concat() method is using object 's' and its data while constructing "s1" object.

### 4) By using factory pattern

\* If method of one java class is returning object of other java class then it is called as factory pattern.

Ex: `StringBuffer sb = new StringBuffer("how are you");`

`String s = sb.substring(6, 3);`

The method of `StringBuffer` class is returning object of `String` class. (factory pattern).

\* If method of one java class wants to return another java class object then these both classes need not be there in inheritance relationship. If they are visible to each other that will be enough.

### 5) By using newInstance() method

```
class c = Class.forName("java.awt.Button");
```

```
Button b = (Button)c.newInstance();
```

\* Here "c" represents the loaded class `java.awt.Button` but it is the object of `java.lang.Class`.  
→ creates and returns the object of `java.awt.Button` class represented with 'b' object

### 6) By using cloning process:

\* The process of creating new object for a java class having existing object data is called as cloning process.

\* programmer can perform cloning on only cloneable objects.

\* The object whose class implements marker interface called `java.lang.Cloneable` is called as cloneable object.

\* The interface whose implementation makes JVM to provide special run time behavior for the objects of class is called as marker interface. (or flag interface).

\* A marker interface can be there with methods or without methods.

Ex: `java.lang.Cloneable`

`java.lang.Serializable`

`java.io.Externalizable`

`java.rmi.Remote`

`javax.servlet.SingleThreadModel`

`java.lang.Runnable`.

- \* When object is created through cloning process constructor will not be executed.
- \* The clone() method of `java.lang.Object` class can perform cloning operation.
- \* When object is created through cloning process the new object and existing object allocates memory separately and acts as two different objects having two different hash codes. So modification done in one object will not reflect in the other object and vice versa.
- \* Every object will be identified through its unique hash code number.

### Example application on cloning

```
// Test.java
```

```
class Test implements java.lang.Cloneable
{
    int a;
    float b;

    public Test()
    {
        System.out.println("Test: 0-param Constructor");
    }

    public Test (int x, int y)
    {
        a = x;
        b = y;
        System.out.println("Test: 2-param constructor");
    }

    public String toString()
    {
        return "a = " + a + "b = " + b;
    }

    public Object myClone() throws Exception
    {
        return super.clone();
    }
}

// Test

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

↳ calls clone method of `java.lang.Object` class

```
//original obj
Test t1 = new Test(10,20);
System.out.println("t1 obj data is:" + t1.toString());
//create obj through cloning
Test t2 = (Test) t1.clone();
System.out.println("t2 obj data is:" + t2.toString());
System.out.println("t1 obj hashCode is:" + t1.hashCode());
System.out.println("t2 obj hashCode is:" + t2.hashCode());
//modify original obj data
t1.a = 30;
System.out.println("t1 obj data is:" + t1.toString());
System.out.println("t2 obj data is:" + t2.toString());
t2.b = 56.4f;
System.out.println("t1 obj data is:" + t1.toString());
System.out.println("t2 obj data is:" + t2.toString());
}
//main
}
// class
>javac mainApp.java  Compilation
>java Main App  Execution
```

## 7) By using deserialization process:

\* the process of writing data of an object to a file is called as serialization.

\* the process of reading data from a file and constructing new object having that data is called as deserialization.

\* When object is created through deserialization constructor will not execute.  
—

\* During java application execution the RAM memory will be utilised to maintain application data. (Input values and results generated by application), due to this we can not use application data after the execution of the application (or) across the multiple executions of the same application.

\* To overcome the above problem store application data in permanent place like files and DB softwares (persistance stores) through persistance operations.

Java app uses (I/o streams) → Files  
(serialization and deserialization)

Java app jdbc concepts → DB software  
(oracle, sybase, mysql)

\* The place where we can store and use data for long time is called as persistance store. Ex: file, database software.

\* A logic that is used to interact with persistance stores and to manipulate the data is called as persistance logic.

\* Performing insert, update, delete and read operations on persistance stores is called as performing persistance store operations.

\* Persistance operations can be called as CURD operations (or) CRUD operations (or) SCUD operations.

C → create (insert), U → update (modify), R → Read (select)  
D → delete (remove).

S → select

C → create

U → update

D → delete

### Limitations with files as persistance stores:

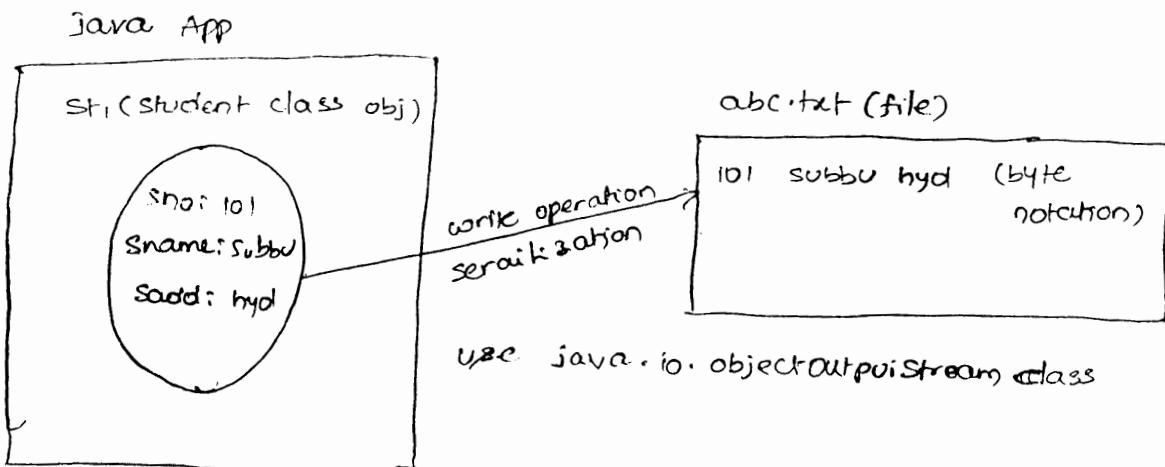
- (1) Can not manage huge amount of data
- (2) NO security
- (3) NO query language support to manipulate the data.
- (4) Delete and update operations are complex operations to perform.
- (5) Relationships are not possible.
- (6) Merging and comparison of data is complex operation.

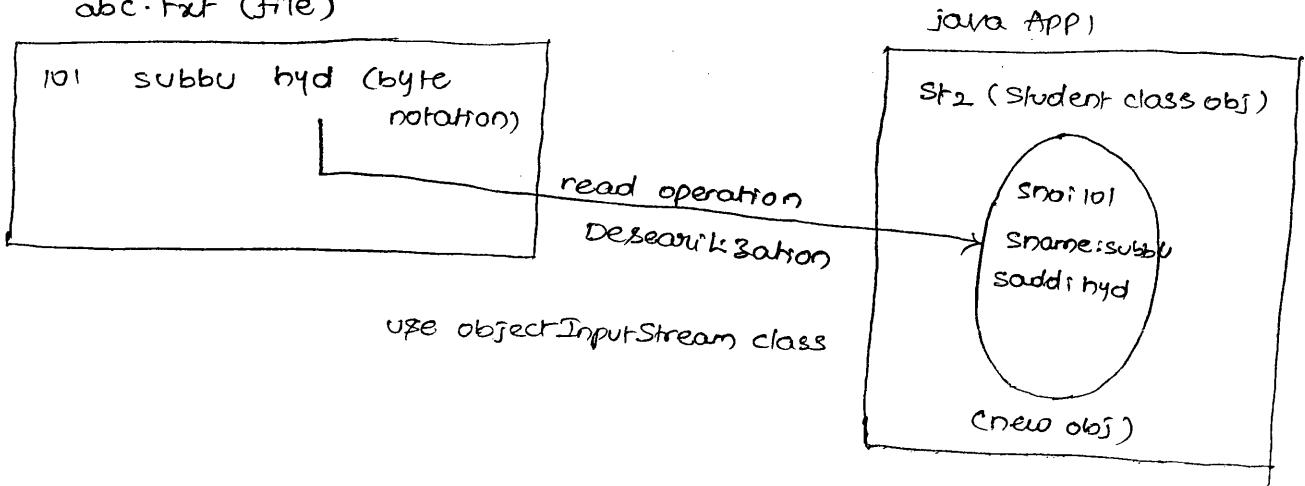
NOTE: To over come above problems use database software as persistance store.

NOTE: In small scale applications like desktop, mobile games use files as persistance store. In medium, large scale applications like bank apps, websites use database software as persistance store.

### Serialization and deserialization:

- \* In small applications the industry generally prefers serialization and deserialization persistance operations on the files.
- \* The process of capturing object data and writing that data into a file is called as serialization.

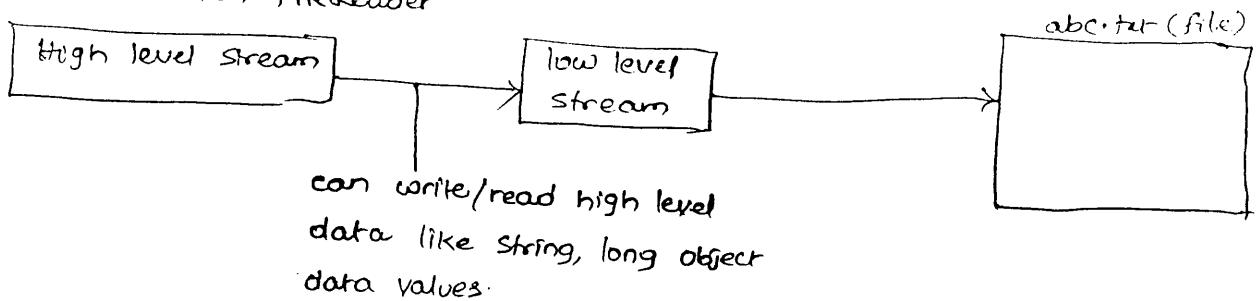
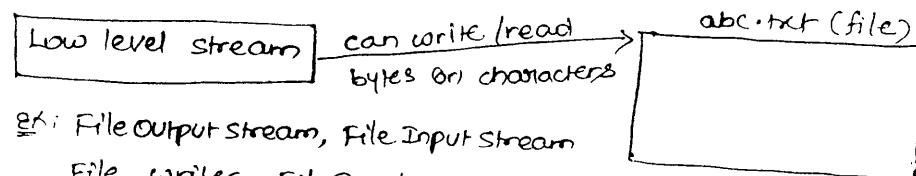




- \* In serialization process object will not be written to a file but the data of the object will be written to a file.
- \* The process of reading data from a file and Constructing new java object having that data is called as deserialization.
- \* A stream is a continuous flow of data that represents communication channel between application and destination resource (file).
- \* Byte streams can write and read bytes.
- \* Character streams can write and read characters.

what is the difference between high level stream and low level stream?

(A)



- Ex:
- DataOutputStream
  - DataInputStream
  - ObjectOutputStream
  - ObjectInputStream

- \* High level streams interacts with files through low level streams and can work with high level data like string values, object values and etc.
- \* Low level streams interacts with files directly and can deal with low level data like characters and bytes.

### Sample Application

- (1) Read sno, name, m<sub>1</sub>, m<sub>2</sub>, m<sub>3</sub> values from key board.
  - (2) calculate total and avg
 
$$\left. \begin{array}{l} \text{total} = m_1 + m_2 + m_3 / 3.0f; \\ \text{avg} = \text{total} / 3.0f; \end{array} \right\} \begin{array}{l} \xrightarrow{\quad} \text{Presentation logic} \\ \text{Business logic} \end{array}$$
  - (3) generate rank for the student → business logic
  - (4) store student details in db table as record → persistance logic
  - (5) display student details and results on monitor → presentation logic.
- \* Presentation logic provides user interface for end user to supply input values and to display results
  - \* Business logic generates results based on given input values (it is main logic of the application)
  - \* Persistance logic manipulate application data in persistance stores
  - \* If entire data of an object is written to a file that is called complete serialization.
  - \* If partial data of an object is written to a file then that is called selective serialization.
  - \* If class is having static or transient variables then objects of that class supports only selective serialization.
  - \* Static member variables allocates memory outside the object so they doesn't participate in serialization.
  - \* Transient member variables allocates memory inside the object but they will be marked for not participating in serialization process.

- \* Programmer can perform persistence operations only on serializable objects.
- \* Java object becomes serializable only when class of that object implements `java.io.Serializable` interface (marker, empty interface).

```
public class student implements java.io.Serializable
{
    int sno;
    String name;
    float avg;
    ...
}
```

objects of this class supports only complete serialization.

```
public class student implements java.io.Serializable
{
    int sno;
    String name;
    static / transient float avg;
    ...
}
```

objects of this class supports only selective serialization.

### Sample code for serialization.

```
FileOutputStream fos = new FileOutputStream ("abc.txt"); // low level stream
ObjectOutputStream oos = new ObjectOutputStream (fos); // high level stream
// create object for student class
Student st1 = new Student (101, "raja", 47.65f);
// Perform serialization
oos.writeObject (st1);
    ↳ captures "st1" object data and writes that data to a file.
// Close streams
fos.close();
oos.close();
```

## Prototype of writeObject() method

public final void writeObject (object obj) throws IOException.

Three important statements.

- (1) If java method parameter type is Java class name then you can call that method by having that class or one of its subclass object as argument value.
- (2) If java method parameter type is abstract class name then you can call that method having one subclass object of that abstract class as argument value.
- (3) If java method parameter type is interface name then you can call that method with one implementation class object of that interface.

## sample code for deserialization

// create highlevel Stream

```
ObjectInputStream ois = new ObjectInputStream (new FileInputStream  
("abc.txt"));
```

// perform de-serialization

object obj = ois.readObject();

// type casting

Student st2 = (Student) obj;

// close streams

ois.close();

→ reads data from the abc.txt file and creates and returns java class object having that data.

## Prototype of readObject() method

Public final Object readObject () throws IOException, ClassNotFoundException.

- (1) If java method return type is Concrete class name then that method can return either that concrete class object or one of its subclass object.

- O (2) If java method return type is interface name then that method returns one implementation class object of that interface.
- O (3) If java method return type is abstract class name then that method returns one subclass <sup>object</sup> of that abstract class.

### Application on serialization and deserialization

#### Resources of the application

student.java

writeTest.java (perform serialization)

readTest.java (Perform deserialization)

// student.java

import java.io.\*;

public class Student implements Serializable

{

    int sno;

    String sname;

    float avg;

    public Student()

{

        System.out.println("student: 0-param constructor");

}

    Student (int x, String y, float z)

{

        sno = x;

        sname = y;

        avg = z;

        System.out.println("student: 3-param constructor");

}

    public void disp()

{

        System.out.println("sno=" + sno + "sname=" + sname + "avg=" + avg);

}

}

```

// writeTest.java

import java.io.*;
public class writeTest
{
    public static void main(String args[])
    {
        try
        {
            // create stream objects
            FileInputStream fos = new FileInputStream("abc.txt");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            // perform serialization
            Student st1 = new Student(101, "anil", 45.78);
            oos.writeObject(st1);
            // close streams
            fos.close();
            oos.close(); System.out.println("serialization is completed");
        } // try
        catch (IOException ioe) // to handle known exception
        {
            ioe.printStackTrace();
        }
        catch (Exception e) // to handle unknown exception
        {
            e.printStackTrace();
        }
    } // main
} // class

```

```

> javac student.java
> javac writeTest.java
> java writeTest

```

NOTE: when serialization is done on the object it not only writes object data to a file but it also writes structure of the class to that file.

\* If application tries to perform serialization on non-serializable object then that the application throws java.io.NotSerializableException

```

Exception.

//readTest.java

import java.io.*;

public class readTest
{
    public static void main(String args[])
    {
        try
        {
            //create stream
            ObjectInputStream ois = new ObjectInputStream(new FileInputStream("abc.txt"));

            //perform deserialization
            Object obj = ois.readObject();

            //type casting
            Student st2 = (Student) obj;

            //display object (st2).data
            st2.display();

            //close stream
            ois.close();
        }
        catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
        catch (ClassNotFoundException cnf)
        {
            cnf.printStackTrace();
        }
    } //main
} //class

```

> javac readTest.java  
> java readTest

\* There is no terminology called selective deserialization. The object data that is return through serialization process must be retrieved through de serialization process.

\* The structure of the class must be same while performing serialization and deserialization operations. otherwise the deserialization

process raises `java.io.InvalidClassException`.

\* In order to send java object over the network it must be designed as serializable object.

\* All wrapper class objects and collection framework data structures are serializable by default.

Elements kept in ArrayList are non-serializable java objects. Can you tell me whether we can send these java objects over the network or not?

No, because ArrayList object and its elements must be there as serializable objects in order to send ArrayList object over the networks.

\* A specification is a document that contains set of rules and guidelines to develop the software or software applications.

\* There are two types of specifications.

(1) open specification (Free to every one)

Ex: JDBC specification, servlet specification and etc..

(2) Proprietary specification (specific to one organization)

Ex: .net specification

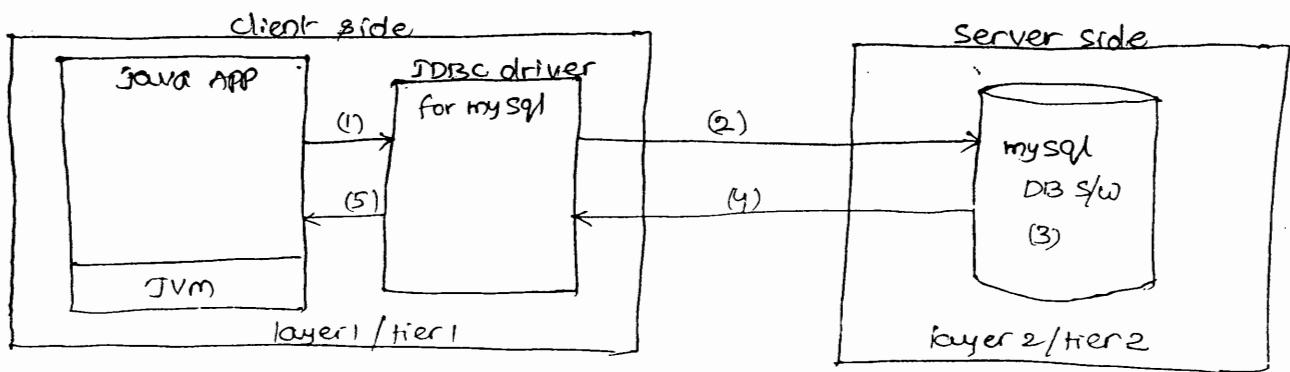
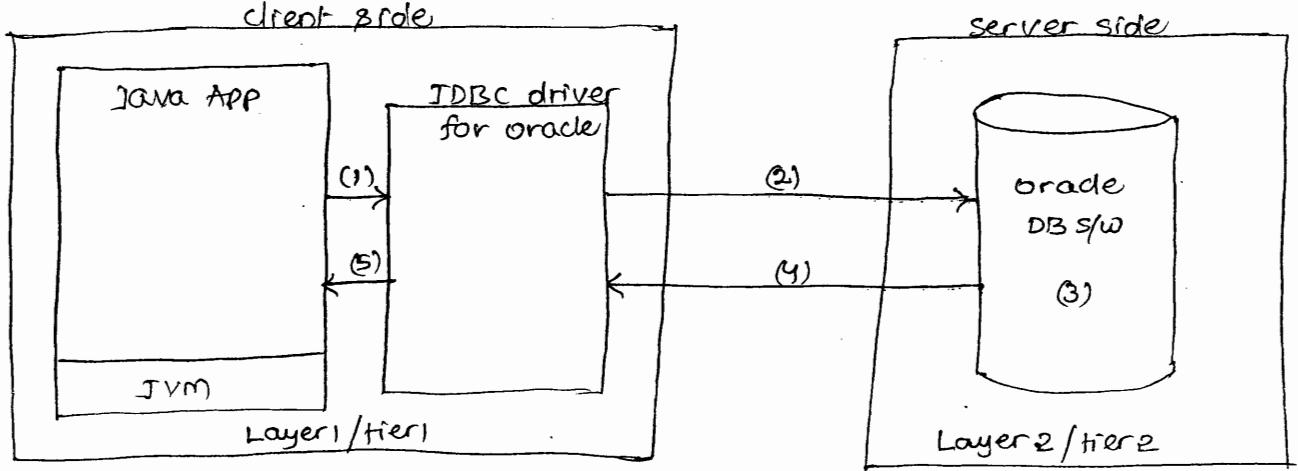
NOTE: Any company can create softwares based on open specification

\* Only the creator of specification is allowed to develop software based on proprietary specification

## JDBC

\* JDBC is an open specification given by Sun Microsystems having rules and guide lines to develop JDBC drivers.

\* JDBC driver is a bridge software between java application and database software. It converts java calls to database calls and viceversa.



NOTE: The above JDBC applications are two tier applications. The two layers of this two tier application can be there either in same computer or in two different computers.

- \* Each JDBC driver is specific to one database software.
- \* We can get JDBC drivers from three parties.

- (1) Sun micro systems
- (2) Database vendor
- (3) Third party vendor.

\* It is recommended to use database vendor supplied JDBC drivers.

\* Non-Java applications use ODBC drivers to interact with database softwares.

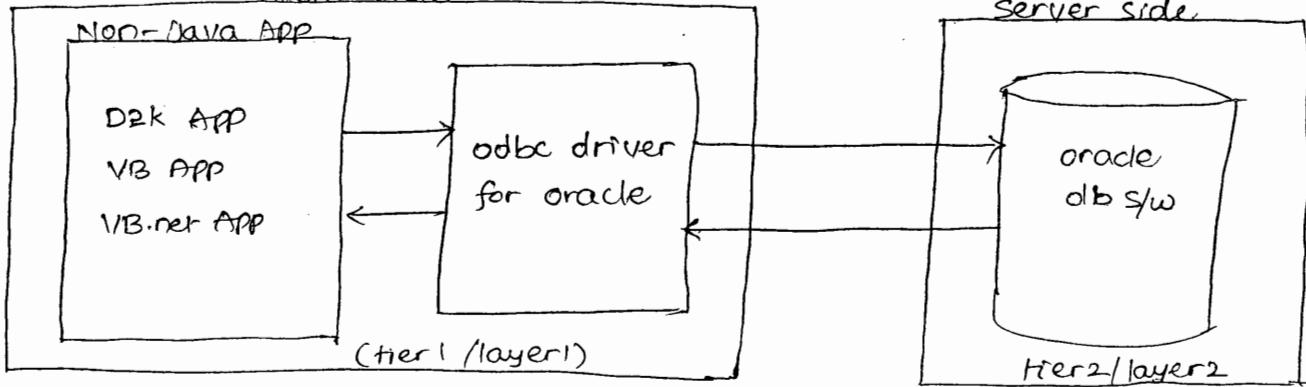
\* Every ODBC drivers is specific to one database software.

What is an ODBC (Open Database Connectivity)?

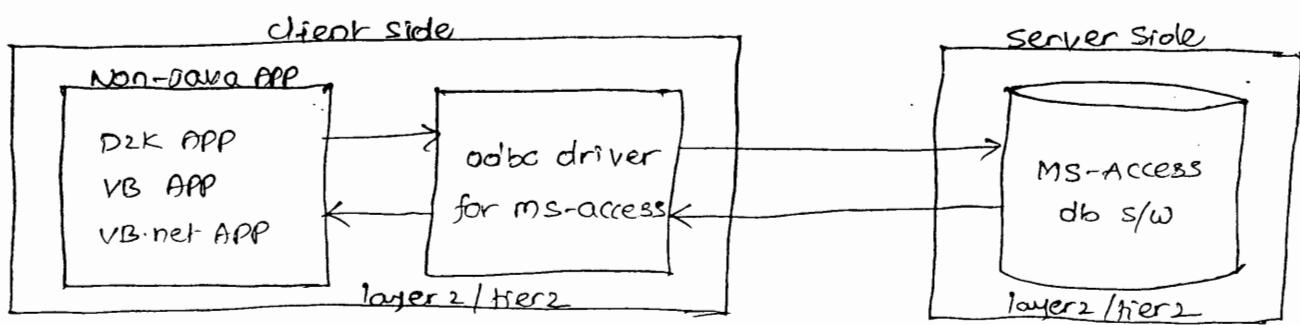
\* ODBC is an open specification that contains given by X-open company that contains set of rules and guide lines to develop

ODBC drivers for different database softwares.

client side



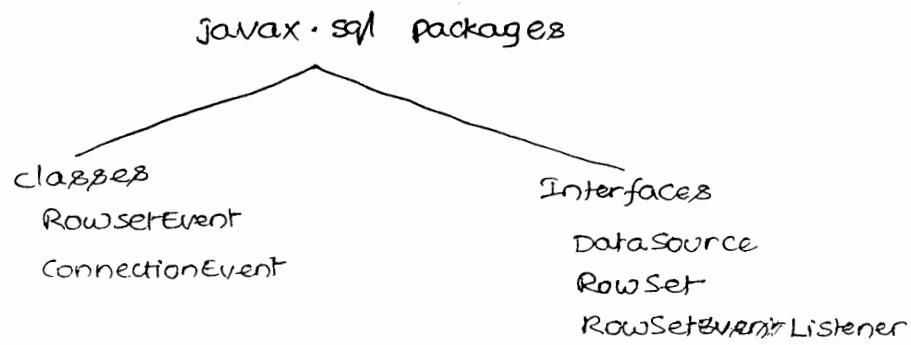
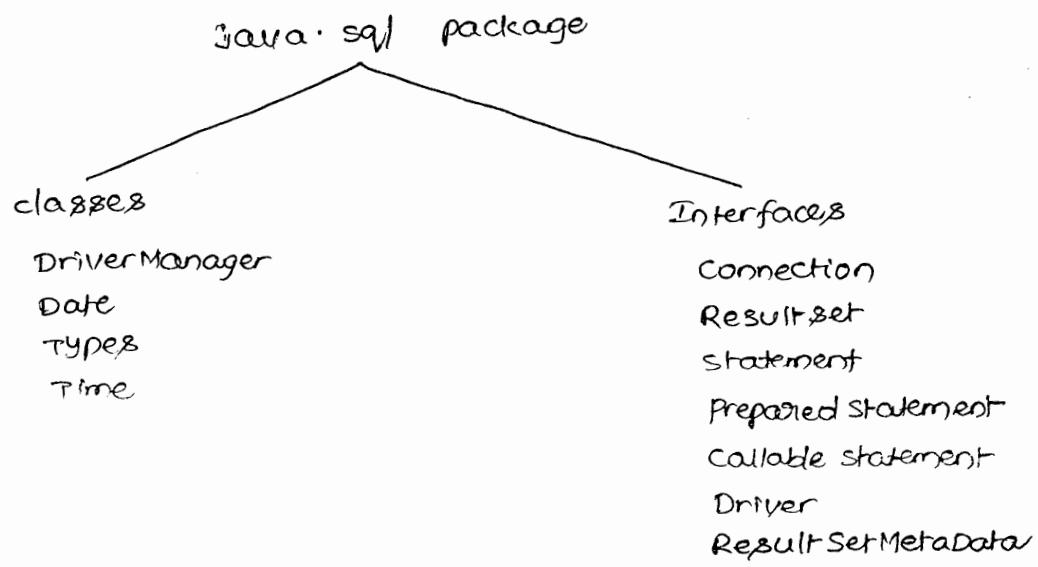
Server side



when all other software technologies are using ODBC drives why java applications need an exclusive JDBC driver to interact with database softwares?

(A) ODBC drivers are given based on ODBC specification. This specification is designed in C language using pointers support. Since Java doesn't support pointers the Java applications need exclusive Java based JDBC drivers to interact with database softwares.

- \* Every specification contains an API representing rules and guidelines.
- \* In Java the abstract methods of API interfaces and abstract classes represent rules whereas the concrete methods of APIs abstract classes, concrete classes represent guidelines.
- \* The JDBC specification related API comes in the form of `java.sql`, `javax.sql` packages.



\* In the above given JDBC API packages concrete methods of abstract, concrete classes represent guidelines to develop JDBC driver. whereas the abstract methods of interfaces and abstract classes represent rules to develop JDBC driver. The vendor company implements these rules and follows these guidelines while developing JDBC driver for each database software.

\* Implementation of all the interfaces that are given in the packages of JDBC API is not the responsibility of programmer, it is the responsibility of vendor company who gives JDBC driver to the market.

\* Each JDBC driver is set of classes implementing various interfaces of JDBC API packages. While defining methods of these interfaces the classes contains logic to interact with specific database software.

\* Since all JDBC drivers are given for different database softwares coming from different vendors are given based on common JDBC specification rules and guidelines given by sunmicrosystems the way we work

with all these JDBC drivers in our Java applications to interact with database softwares is going to be much similar.

sample code to understand JDBC specification, JDBC driver implementation and utilizing JDBC driver from java applications

Assume this is the JDBC specification

MyConnect.java (Interface)

```
public interface MyConnect
{
    public void getConnection(); //rule
}
```

Assume this is JDBC driver for oracle (Vendor1)

```
public class OracleConnect implements myConnect
```

```
{
    public void getConnection()
    {
        ----- logic to interact with oracle db s/w
    }
}
```

Assume this is JDBC driver for MySQL (Vendor2)

```
public class MySQLConnect implements myConnect
```

```
{
    public void getConnection()
    {
        ----- logic to interact with MySQL db s/w
    }
}
```

Application Development

APP1 (Java application)

activates JDBC driver for oracle

```
call getConnection(); // makes APP1 talking with oracle db s/w
```

APP2 (Java application)

activates JDBC driver for mysql

```
call getConnection(); // makes APP2 talking with MySQL db s/w
```

NOTE: In the above sample code App1, App2 applications are using two different JDBC drivers in much similar manner to interact with two different database softwares.

\* Generally vendor companies supply their JDBC drivers in the form of jar files.

\* To get the source code from Compiled code use Java decompiler.

Ex: DT DeCompiler (separate installation)

\* Even though every JDBC driver software contains lots of classes but the JDBC driver will be identified through its driver class name.

\* The java class that implements java.sql.Driver interface directly or indirectly is called as JDBC driver class.

\* To activate JDBC driver from our java applications we need to work with the driver class name of that JDBC driver.

\* JDK software gives one built-in JDBC driver. Its driver class name is sun.jdbc.odbc.JdbcOdbcDriver

↳ package      ↳ Driver class implementing java.sql.Driver interface

\* Oracle database software gives one built-in JDBC driver. Its driver class name is oracle.jdbc.driver.OracleDriver

↳ package      ↳ Driver class

\* we can expect odbc drivers based on odbc specification from three parties.

(1) X-open Company

(2) DB Vendor

(3) Third party vendor

NOTE: The windows XP OS installation gives us multiple odbc drivers.

\* Every odbc driver is identify through its DSN. There is a possibility of creating three types of dsn's.

(1) User DSN → specific to currently logged in windows user

(2) System DSN → visible all the windows users of a machine.

(3) File DSN → sharable in the network machines.

Procedure to create Microsoft ODBC driver for oracle:

start → settings → control panel → performance and maintenance →

administrative tools → datasources → user dsn → add → microsoft odbc

for oracle → finish → data source name xyz    username scott

server [optional] for local or personal oracle software installation, mandatory

for the centralized installation of oracle software (gather dns string)

→ ok → ok

\* There are four different methodologies or mechanisms to develop JDBC drivers based on the rules and guidelines of JDBC specification. They are.

(1) Type 1

(2) Type 2

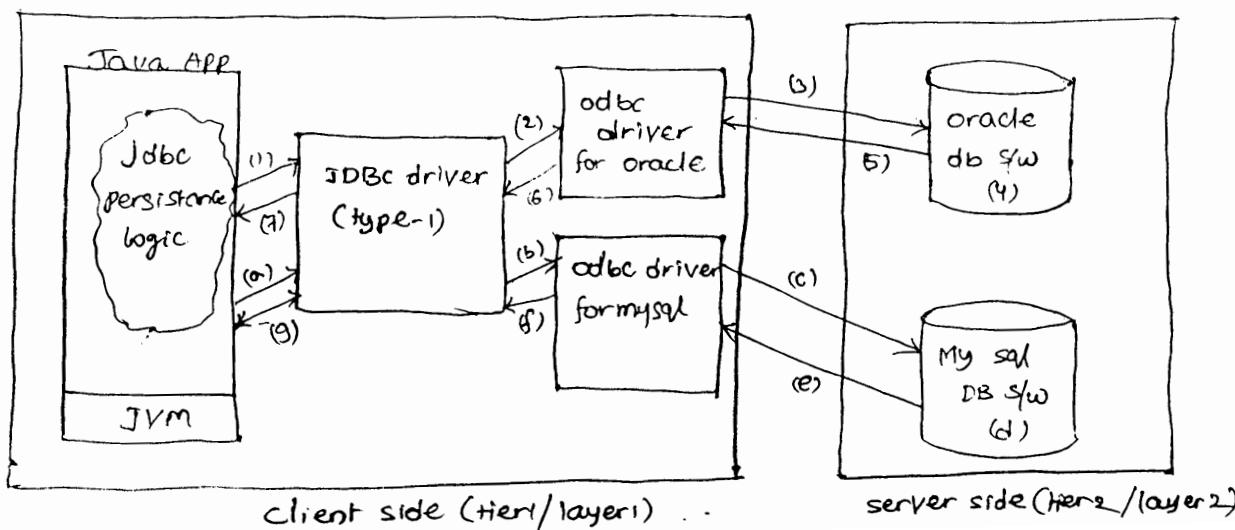
(3) Type 3

(4) Type 4

\* Every mechanism talks about architecture or plan to develop JDBC drivers.

\* The type 1 mechanism based drivers are given to take the support of ODBC drivers while interacting with database softwares. In this process the type 1 driver takes the support of native code to communicate with ODBC drivers.

TYPE1 driver architecture (partial)



- \* JDBC driver type-I is not specific to any database software because it won't interact with database software directly. It interacts with database softwares by using the database software specific odbc driver.
- \* only sun micro system is supplying type-I mechanism based JDBC drivers as built-in JDBC driver of JDK software.
- \* JDK software supplies a basic and built-in service called driver manager service to manage set of JDBC drivers and to establish the connections with database softwares by using JDBC drivers.
- \* Every JDBC driver must be registered with DriverManager service. For this create JDBC driver class object and keep that object in driver manager service.
- \* Our Java applications can activate and work with DriverManager service by dealing with java.sql.DriverManager class.

JDK's built-in driver is Type I driver

this driver's class name is: sun.jdbc.odbc.JdbcOdbcDriver

To register the above typeI driver with DriverManager service

```
//create object for JDBC drivers class
```

```
sun.jdbc.odbc.JdbcOdbcDriver obj = new sun.jdbc.odbc.JdbcOdbcDriver();
```

```
//register JDBC driver with DriverManager service,
```

```
DriverManager.registerDriver(obj);
```

\* The JDBC API of JDBC specification given by SunMicro Systems will be used in two different angles.

(1) As rules and guide lines by vendor companies to develop JDBC drivers.

(2) As base for the programmer to activate JDBC driver and to develop JDBC persistence logic from Java applications. This persistence logic is responsible to manipulate data base data (CURD operations).

\* One JDBC driver typeI can interact with multiple odbc drivers of different db s/w's.

## Standard steps to develop jdbc code in our java Application

- (1) register jdbc driver with DriverManager service
- (2) Make DriverManager service to establish connection with DB SW.
- (3) create statement object.
- (4) Use statement object to send sql queries to database software and to execute them in database software.
- (5) Use statement object to gather results from database software.
- (6) Process the results
- (7) close connection with database software

\* Statement object acts as courier service or vehicle to send and receive inputs and outputs between java application and database software. This is base for the programmer to manipulate the database data from java applications.

- \* Connection between java application and database software represents the communication channel between java application and database software.
- \* To establish this connection the driver manager service uses an appropriate one of the registered JDBC driver.
- \* The code that is not written in java but invokable from java applications is called as native code.
- \* JDBC driver type 1 internally uses this native code to interact with ODBC drivers.
- \* Ex: Except type 1 mechanism drivers no other mechanism drivers (type-2, type-3, type-4) use ODBC drivers support to interact with database softwares.

Write a java application to establish the connection with oracle database software by using JDBC driver type-1 (Built-in JDBC driver of Jdk software)

The software setup that is required

Jdk any version (Jdk 1.6)

oracle db sw (any version)

dsn created for microsoft odbc driver for oracle

//connTest.java

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

```
public class ConnTest
```

```
{
```

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

```
{
```

```
    //create JDBC driver class object (Type-1 driver related)
```

```
    sun.jdbc.odbc.JdbcOdbcDriver obj= new sun.jdbc.Odbc.Jdbc  
        OdbcDriver();
```

```
    // register jdbc driver with driverManagerservice
```

```
    DriverManager.registerDriver(obj);
```

```
    //Establish connection with database software . Protocol
```

```
    Connection con = DriverManager.getConnection ("jdbc:odbc:
```

subname(dsn)      oradsn", "scott", "tiger");  
                        ↑dburl      ↓      ↓      ↓  
                        db user   db password

```
if (con == null)
```

```
    System.out.println ("connection is not established");
```

```
else
```

```
    System.out.println ("connection is established");
```

```
} //main
```

```
} //close
```

```
>javac ConnTest.java
```

```
>java ConnTest
```

\* Based on the protocol specified in the url of drivermanger.

getconnection() method the Drivermanager service picks up an appropriate JDBC driver even though more JDBC drivers are registered with DriverManagerservice . The DriverManager service uses that selected/picked up driver to establish the connection with database software .

- \* JDBC:ODBC protocol makes DriverManager service to pickup the registered type-1 JDBC driver.
- \* JDBC:ODBC is application level protocol
 

↓
T
→

main protocol      sub protocol
- \* Protocol is a set of rules followed by two parties who participates in communication.
- \* Network level protocol contains rules to get communication between two physical computers. Ex: TCP/IP
- \* Application level protocol contains rules to get communication between two softwares or software applications. Ex: HTTP, JDBC:ODBC
- \* JDBC:ODBC is the application level protocol containing set of rules to get communication between DriverManager service and database software.
- \* Most of the application level protocols can run on the top of network level protocols.

```

Connection con = DriverManager.getConnection("Jdbc:odbc:oradsn", "scott", "tiger");
          ↓
  JDBC Connection
  object representing
  connectivity with
  database software

```

\* In the above statement how can you say "con" is an object when java.sql.Connection is an interface?

- A) In the above statement con is object of underlying JDBC driver supplied java class that implements java.sql.Connection Interface. To know that class name

```
System.out.println("Con obj class name is:" + con.getClass());
```

In type-1 driver con object class name is "sun.jdbc.odbc.JdbcOdbcConnection" and this class implements java.sql.Connection Interface indirectly as shown below.

java.sql.Connection (Interface)

↑ Extends

sun.jdbc.odbc.JdbcOdbcConnectionInterface (Interface)

↑ Implements

sun.jdbc.odbc.JdbcOdbcConnection (class)

\* class "JdbcOdbcConnection" is implementing java.sql.Connection interface.

NOTE: DriverManager.getConnection() method call returns object of a Java class that implements java.sql.Connection Interface.

\* when interface reference variable points to its implementation class object then that reference variable can also be called as indirect object of implementation class.

Connection con = DriverManager.getConnection ("jdbc:odbc;oradsn");

\* In the above statement con is the reference variable of java.sql.ConnectionInterface pointing to object of a class that implements java.sql.ConnectionInterface; so "con" also becomes object of that class.

In our JDBC code why we are not specifying class name of JDBC connection object (con) ?

(A) The class name of connection object will change based on the JDBC driver we use so if that class name is specified our JDBC code becomes specific to one JDBC driver so we never specify that class name to write JDBC code in a common way for all JDBC drivers.

Rewriting ConnTest.java

```
// ConnTest.java  
import java.sql.*;  
public class ConnTest  
{  
    public static void main (String args[])  
    {  
    }
```

```

//load jdbc driver class
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//here type1 driver will be registered with DriverManager
//service internally

//establish connection with db sw
Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                              "scott", "tiger");
if (con == null)
    System.out.println("Connection is not established");
else
    System.out.println("Connection is established");
}
}

```

(Q) In the above code Class.forName(-) method call just keeps makes JVM to load given JDBC driver class. How can you say JDBC driver is registered with DriverManager service in that process?

(A) Class.forName(-) makes JVM to load sun.jdbc.odbc.JdbcOdbcDriver class → static block of this driver class executes automatically → This static block contains logic to create object of JDBC driver class and to register that object with DriverManager service by calling DriverManager.registerDriver() method. In the static block of Sun.jdbc.odbc.JdbcOdbcDriver class we can see the following code registering with type-1 driver with DriverManager service.

```

static
{
    -----
    -----
    //creating type-1 jdbc driver class object
    JdbcOdbcDriver jdbcodbcdriver = new JdbcOdbcDriver();
    try
    {
        //registering type-1 driver with DriverManager service
        DriverManager.registerDriver(jdbcodbcdriver)
    }
}

```

```
o     catch (SQLException sqlexception)
```

```
o     {  
o         ---  
o     }
```

```
o }
```

NOTE: Almost every vendor supplied jdbc driver related jdbc driver class contains the above given static blocks. Registering ~~this~~ its respective jdbc driver with drivermanager service.

(Q) How many ways are there to register jdbc driver with DriverManager service?

(A) (1) 

```
sun.jdbc.odbc.JdbcOdbcDriver obj = new sun.jdbc.odbc.JdbcOdbcDriver();  
Drivermanager.registerDriver(obj);
```

The above code registers jdbc driver (same) twice with Driver manager service.

- (i) to static block of driver class when driver class object is created.
- (ii) Explicitly when programmer calls `Drivermanager.registerDriver()`. It is not recommended process.

(2) 

```
sun.jdbc.odbc.JdbcOdbcDriver obj = new Sun.jdbc.Odbc.JdbcOdbcDriver();
```

Here jdbc driver will be register Drivermanager service only once but programmer creates one explicit unused object for driver class. It is not recommended approach.

(3) 

```
DriverManager.registerDriver (new sun.jdbc.odbc.JdbcOdbcDriver());
```

Limitations are same as approach no.1

→ Anonymous object

(4) 

```
Class.forName ("sun.jdbc.odbc.Jdbc Odbc Driver");
```

Static block of driver class registers jdbc driver with Driver manager service only once.

It is recommended approach.

(5) Class.forName(args[0]);

```
// ConnTest.java
import java.sql.*;
public class ConnTest
{
    public static void main (String args[])
    {
        //load jdbc driver class
        Class.forName (args[0]);
        //establish connection with db s/w
        Connection con = DriverManager.getConnection ("jdbc:odbc:oradsl",
            "scott", "tiger");
    } //main
} //class

>javac ConnTest.java
>java ConnTest sun.jdbc.odbc.JdbcOdbcDriver
                ↴ args[0] value
```

same as approach no. 4 but allows to pass JDBC driver class name from outside the application dynamically at runtime. It is recommended approach.

(6) we can pass input value to java application as system property value by using java -D option. To read this value our application can use `System.getProperty()` method.

```

    } // main
}

>javac ConnTest.java
>java -Dmy.driver = sun.jdbc.odbc.JdbcOdbcDriver ConnTest
    ↴ system property name and value

```

Recommended to use same as approach no. 5.

- (7) The DriverManager class attempts to load the driver classes automatically which are referenced in the system property called "jdbc.drivers"

```

//ConnTest.java
import java.sql.*;
public class ConnTest
{
    public static void main(String args[]) throws Exception
    {
        //establish connection with db s/w
        Connection con = DriverManager.getConnection ("jdbc:odbc:oradsn",
                                                    "scott","tiger");
    }
}

```

```

} //main
} //class
>javac ConnTest.java
>java -Djdbc.odbc.JdbcOdbc
>java -Djdbc.drivers = sun.jdbc.odbc.JdbcOdbcDriver ConnTest
    ↴ fixed
        property name

```

The above process makes current machine remembering registered jdbc driver for long time.

- (8) In the process of loading subclass the JVM automatically loads superclass. In this process the static blocks of both subclass and superclass will be executed automatically. The below process is not recommended if makes our class not extending from other useful classes like Thread, Frame, Applet and etc.

```

// ConnTest.java
import java.sql.*;
public class ConnTest extends sun.jdbc.odbc.JdbcOdbcDriver
{
    public static void main (String args[])
        throws Exception
    {
        // Establish connection with DB S/W
        Connection con = DriverManager.getConnection(

```

\* According to DBMS the sql queries are

- (1) DML queries (insert, update, delete queries)
- (2) DRL queries (select queries)
- (3) DDL queries (createtable, alter table and etc...)
- (4) TCL queries (commit, rollback, savepoint and etc...)

\* According to jdbc persistance logic there are two types of sql queries

- (1) select sql queries (DRL queries)
- (2) Non-select sql queries (DDL, DML, TCL queries)

\* select sql query execution returns bunch of records that are selected from database table, whereas non-select queries (like DML queries) return a numeric value representing no. of records that are effected because of query execution.

\* Our jdbc persistance logic uses executeUpdate() to send and execute non-select queries in database software. Similarly it uses executeQuery() method to send and execute select queries to db s/w.

### Prototype of executeQuery (-) method

public Resultset executeQuery (String query) throws SQLException.

\* executeQuery (String query) method returns jdbc resultset object having set of records collected from database software after executing select sql query.

### Prototype of executeUpdate (-) method

public int executeUpdate (String query) throws SQLException

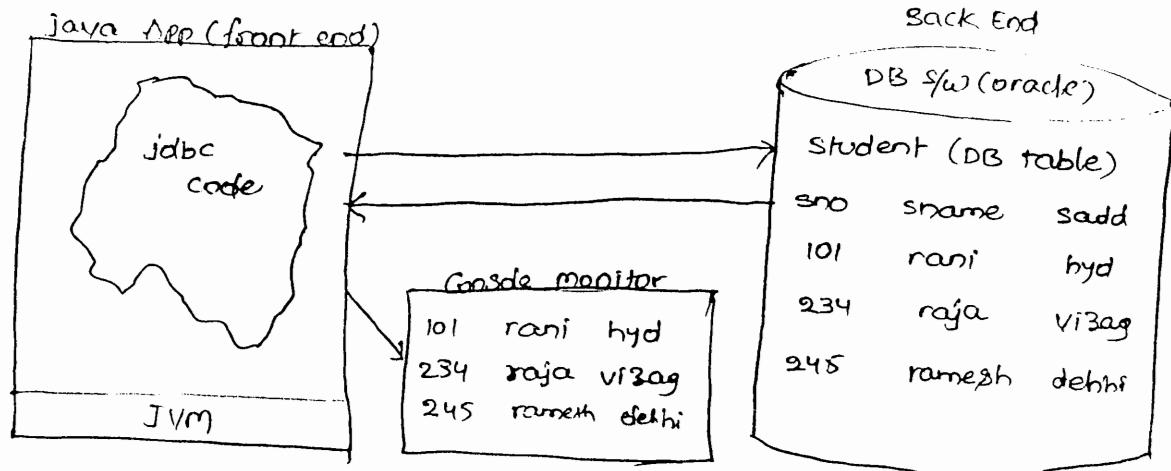
\* This method returns a numeric value representing no. of records that are effected after executing non-select sql query in database software.

\* The above given two methods are declared in java.sql.Statement Interface.

\* Since the end users like civil engineers, bank employees and etc. can not work with database software directly by using sql queries. They expect ~~GUI~~ GUI front end applications as front end to perform various operations on back end called d/b s/w.

\* While developing these front end applications in java environment we use jdbc support to interact with database software and to manipulate database data.

Write a java application to read the records of database table from java application



Step(1): Make sure that oradsn is available as dsn for odbc driver for oracle.

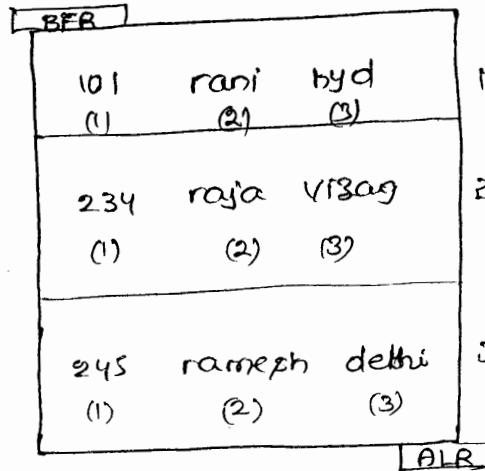
Step(2): keep student database table ready in database software (oracle) with records.

Step(3): Develop JDBC code based application using type1 driver as shown below.

// selectTest.java

```
import java.sql.*;
public class selectTest
{
    public static void main (String args[])
        throws Exception
    {
        // load jdbc driver class
        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
        // establish connection with database software
        Connection con = DriverManager.getConnection ("jdbc:odbc:oradsn",
                                                    "scott", "tiger");
        // create jdbc statement object
        Statement st = con.createStatement ();
        // send and execute sql query in database software and
        // gather the results.
        ResultSet rs = st.executeQuery ("select * from student");
        // process the result
        while (rs.next () == true)
        {
            System.out.println (rs.getInt (1) + " " + rs.getString (2) +
                                " " + rs.getString (3));
        }
        // close jdbc objects
        rs.close ();
        st.close ();
        con.close ();
    }
}
```

rs (result set object)



\* Every result set object contains two positions

(1) BFR (Before First record)

(2) ALR (After Last record)

\* By default record pointer / cursor resides in BFR position.

\* Records in result set and values in each record of result set will be stored having one based index.

\* rs.next() method moves the record pointer from current position to next position. If the next position is valid position (record) then this method returns true. If the next position is ALR or BFR position then this method returns false.

\* To read each record values from JDBC result set object we can use getXXX() methods, with column name (or) column index.

getXXX(-)

getString(-), getFloat(-), getInt(-), getDouble(-), getLong(-), getDate(-) and etc.

Prototype of getXXX(-) methods:

Public YYY getXXX (String columnname / int column index) throws  
SQLException.

- \* It is always recommended to close the jdbc objects in the reverse order of their creation.
- \* If you don't close jdbc objects, at the end of their utilization then they will be closed automatically and abnormally at the end of the application's execution.
- \* The jdbc applications can not read uncommitted data from the database tables.
- \* JDBC statement object means it is the object of a jdbc driver supplied java class implementing java.sql.StatementInterface directly or indirectly.
- \* JDBC resultset object means (or) it is the object of a jdbc driver supplied java class implementing java.sql.ResultSet interface directly or indirectly.
- \* All jdbc objects of jdbc programming are objects of jdbc driver supplied java classes implementing various interfaces of jdbc API but programmer never specifies these class names in his jdbc programming because their names will be change based on the jdbc driver.
- \* When interface reference variable points to its implementation class object then reference variable is also becomes object of implementation class with limitations.

```

interface ABC
{
    public void x();
}

public class Demo implements ABC
{
    public void x()
    {
        System.out.println("Demo: x()");
    }

    public void y()
    {
        System.out.println("Demo: y()");
    }
}

```

```
public static void main (String args[])
{
    ABC ab = new Demo();
    ab.x(); // valid
    ab.y(); // invalid
    System.out.println("class name of ab is " + ab.getClass());
}
```

↳ gives Demo class.

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

- \* In the above statement a) `createStatement()` is the method declared in `java.sql.Connection` interface (As the rule of JDBC specification).
- b) The JDBC driver supplied java class that implements `java.sql.Connection` interface contains implementation for this `createStatement()` method (In type-1 driver that class name is `sun.jdbc.odbc.JdbcOdbcConnection`).
- c) The definition of `createStatement()` method contains logic to create and return object of a java class (JDBC driver supplied) that implements `java.sql.Statement` interface (In type-1 driver that class name is `sun.jdbc.odbc.JdbcOdbcStatement`).

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

- \* `getconnection` is the static method of `DriverManager` class. This method definition returns object of JDBC driver supplied java class implementing `java.sql.Connection` interface as JDBC connection object (JDBC driver supplied class). (In type-1 driver class is `sun.jdbc.odbc.JdbcOdbcConnection`).

#### >SelectTest.java

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

```

public class Test1
{
    public static void main(String args[]) throws Exception
    {
        -----
        ----- } Logic to create jdbc connection
        ----- object

        //read input values from key board.

        Scanner sc = new Scanner(System.in);
        System.out.print("enter student no:");
        int no = sc.nextInt();

        //create jdbc statement object
        Statement st = con.createStatement();
        //send and execute sql query in db sw and gather results
        Resultset rs = st.executeQuery("select * from student
                                         where sno >= " + no);

        //Process the result (Resultset obj)
        while(rs.next() != false)
        {
            System.out.println(rs.getInt("sno") + " " + rs.getString("sname") + " "
                               + rs.getString("saddr"));
        }

        //close jdbc objects
        rs.close();
        st.close();
        con.close();

    } // main
} // class

```

\* while retrieving column values from the records of jdbc Resultset object use those column indexes in the order column values are stored in the records of jdbc Resultset object. Don't use those column indexes in the order columns are there in the database table.

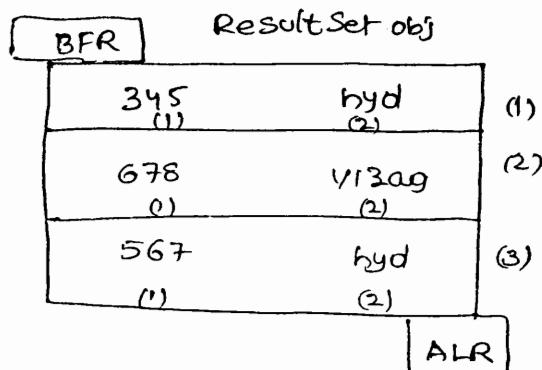
### Example code :

```

statement st = con.createStatement();
ResultSet rs = st.executeQuery("Select sno, saddr from student");
// Process the resultset object
while (rs.next() != false)
{
    System.out.println(rs.getInt(1) + " " + rs.getString(2));
}

```

↳ gives sno column values
↳ gives saddr column values



Selecting specific column values from database table with multiple conditions

//read input values from keyboard

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

SOP("Enter starting range of student no:");

```
int stno = sc.nextInt();
```

```
sop("enter ending range of student no:");
```

```
int endno = sc.nextInt();
```

//create jdbc statement obj

Statement st = Con.createStatement();

Result set rs = st. executeQuery("select \* from student")

student where sno >= " + start" and sno = " + endno);

```
//process the result set object  
while (rs.next() != false)
```

```
{  
    sop(rs.getString(1) + " " + rs.getString(2));  
}  
}      ↓      ↓  
gives saddr values   gives sname values
```

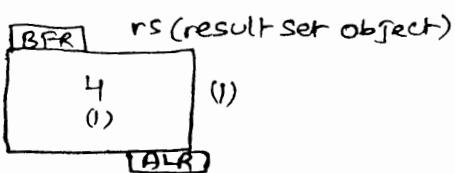
\* while framing SQL queries with string input values the programmer needs to be little bit careful

Example Code:

```
//read input values from keyboard  
Scanner sc = new Scanner(System.in);  
  
sop("Enter begining char or chars of student name to perform  
search operation");  
  
String cond = sc.next(); // gives "char:r"  
  
//converting input value as required for sql condition  
Cond = "%" + Cond + "%"; // r becomes 'r%'  
  
//create jdbc statement object  
Statement st = con.createStatement();  
  
ResultSet rs = st.executeQuery("Select * from student where  
sname like "+Cond);  
  
//process the resultset object  
  
while (rs.next() != false)  
{  
    sop(rs.getString(1) + " " + rs.getString(2) + " " + rs.getString(3));  
}
```

\* Aggregate functions retrieves data from database table as not specific to any record of the table.

Example to send and execute sql aggregate function based query



```

    //create jdbc statement object
    Statement st = con.createStatement();
    ResultSet rs = new ResultSet;
    ResultSet rs = st.executeQuery("select count(*) from
                                    student");
    //process the jdbc Resultset object
    if (rs.next())
    {
        int cnt = rs.getInt(1);
        //int cnt = rs.getInt("Count(*)");
        System.out.println("Count of records: " + cnt);
    }

```

\* Once Resultset object is closed we can not retrieve data from Resultset object.

\* When JDBC objects are closed by calling close() method all the database resources that are associated with these objects will be released immediately.

\* To solve unsupported class version exception problem that raises after installing Oracle software make sure that java-home\bin directory is kept as first value of path environment variable.

my computer → properties → advanced tab → env variables

variable name: Path

Value: D:\Java\jdk 1.6.0\_11\bin; D:\oracle\product\10.2.0\

↳ java-home

↳ db-1\bin; .

↳ oracle-home current  
directory

OK → OK → OK

Example code to send and execute multiple aggregate functions based  
select sql query

```

    //create jdbc statement object
    Statement st = con.createStatement();

```

```

ResultSet rs = st.executeQuery("select count(*), max(sno),
                                min(sno), avg(sno), sum(sno) from student");

// process the jdbc result-set object

if (rs.next())
{
    System.out.println("Count of records: " + rs.getInt(1));
    System.out.println("max val of sno: " + rs.getInt(2));
    System.out.println("min val of sno: " + rs.getInt(3));
    System.out.println("sum of sno column: " + rs.getInt(4));
    System.out.println("avg of sno col value: " + rs.getFloat(5));
}

```

BFR	rs (JDBC resultset obj)
4 567 101 1115 278.5	(1) (2) (3) (4) (5)
	[ALR]

Example code to execute sub query (Gives that student record who is having max sno value)

```

// create jdbc statement object
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from student where
                                sno = (select max(sno) from student)");

// process the jdbc resultset object

if (rs.next())
{
    System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
}
else
    System.out.println("no record found");

```

BFR	rs (result-set obj)
567 ramesh hyd	(1) (2) (3) (4)
	[ALR]

Example code that selects the data from multiple columns of multiple database tables.

```
//create jdbc statement object  
Statement st = con.createStatement();  
  
ResultSet rs = st.executeQuery ("select empno, ename, dname,  
loc from emp, dept");  
  
//process the resultset object  
while (rs.next())  
{  
    System.out.println(rs.getInt("empno") + " " + rs.getString("ename") + "  
    " + rs.getString("dname") + " " + rs.getString("loc"));  
}
```

BFR					rs (jdbc resultset obj)
567	Xyz	Sales	USA	(1)	
(1)	(2)	(3)	(4)		
ALR					

write a java application by using jdbc driver type-1 to delete the records of student table based the given student no.

```
// DeleteTest.java (use executeUpdate() method)  
import java.sql.*;  
import java.util.*;  
  
public class DeleteTest  
{  
    public static void main (String args[]) throws Exception  
    {  
        // read input values from keyboard  
        Scanner sc = new Scanner (System.in);  
        int no = sc.nextInt();  
  
        // register jdbc driver with DriverManager service  
        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
```

```

//establish connection with DB software
Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                             "scott", "tiger");
//create jdbc statement object
Statement st = con.createStatement();
//send and execute query in db software
int res = st.executeUpdate("delete from student where
                           sno = " + sno);
//process the result
if (res == 0)
    System.out.println("no record found to delete");
else
    System.out.println(res + " no. of records are deleted");
//close jdbc objects
st.close();
con.close();
}

```

\* Java application sends and execute non-select queries on database software by enabling autoCommit mode. we can disable this autoCommit mode from our java application by using `con.setAutocommit(false)` method.

\* After disabling autoCommit mode if java application sends and executes non select sql queries in database software then use `con.commit()` method to commit the query execution results (or) use `con.rollback()` method to rollback the query execution results.

### Example code

```

// disable auto Commit mode
con.setAutoCommit(false);
//create jdbc statement object

```

```

    Statement st = con.createStatement();
    //Send and execute sql query in db s/w
    int res = st.executeUpdate("delete from student where sno="
                               +no);
    //rollbacks or Commit
    con.rollback(); //or con.commit();

```

\* The above code demonstrates TCA operations.

(Q) When all other SQL queries are going to database software by using JDBC statement object why commit and rollback instructions should be given to database software by using JDBC connection object?

A) Statement object deals with single query execution at a time. But commit and rollback operations should be performed with respect to multiple SQL queries execution so we use JDBC connection object which represents whole database to pass the commit and rollback instructions.

The SQL queries whose execution can generate results should be given to database software by using JDBC statement object. The instructions whose execution do not return results like commit, rollback should be given to database software using JDBC connection object.

Write a Java application to insert the record into database table by gathering values from keyboard.

```

// insertTest.java
import java.sql.*;
import java.util.*;
public class insertTest
{
    public static void main(String args[])
    {
        //read input values from keyboard
        Scanner sc = new Scanner(System.in);

```

```
System.out.print("Enter student no.");
int no = sc.nextInt();
System.out.println("Enter student name");
String name = sc.next(); // raja
System.out.println("Enter student address");
String addr = sc.next(); // hyd
// Convert input values as required for sql query
name = "'" + name + "'"; // raja becomes 'raja'
addr = "'" + addr + "'"; // hyd becomes 'hyd'
// create jdbc connection object
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                            "scott", "tiger");
// create jdbc statement object
Statement st = con.createStatement();
// frame example sql query with direct values
// insert into student values (101, 'raja', 'hyd');
// frame query with variables
String query = "select insert into student values (" + no + ", " + name +
               ", " + addr + ")";
System.out.println(query);
// send and execute sql query in db s/w
int result = st.executeUpdate(query); // non-select
// process result
if (result == 0)
    System.out.println("Record insertion is failed");
else
    System.out.println("Record insertion is done");
```

```
//close jdbc objects  
st.close();  
con.close();  
}  
//main  
}  
//class
```

## Industry coding standards

- \* Don't declare the exception to be thrown using throws statement.
- \* Always catch and handle the exceptions with try and multiple catch blocks.
- \* Write comment for every line.
- \* To avoid NullPointerException call method on java object after ensuring that object is not holding null value. <sup>close</sup> These jdbc objects <sup>(or)</sup> are any other stream objects in finally block because that block executes irrespective of the exception that is raised and gives the guarantee of closing stream objects in any situation.

## Write a java application to update student details based on student no.

// updateTest.java (with coding standards)

```
/* APP to update student details  
 * version : 1.0  
 * author : Team-S */  
  
import java.sql.*;  
import java.util.*;  
  
public class UpdateTest  
{  
    public static void main(String args[])  
    {  
        Connection con = null;  
        Statement st = null;  
        Scanner sc = null;
```

```

try
{
    //read input values from keyboard
    sc = new Scanner(System.in);
    int sno=0;
    String name=null;
    String address=null;
    if (sc!=null)
    {
        System.out.println("Enter existing student number");
        sno = sc.nextInt();
        System.out.println("Enter new name");
        name = sc.next();
        System.out.println("Enter new address");
        address = sc.next();
    } //if

    //create jdbc con obj
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    con = DriverManager.getConnection("jdbc:odbc:oradsn","scott","tiger");
    // create statement object
    if (con != null)
        st = con.createStatement();
    // frame example query with direct values
    // String qury = "update student set sname='maharaja', sadd='vizag'
    //                 where sno=101";
    string qury = "update student set sname='"+fname+"', sadd='"+address+
    "+'"' where sno='"+no;
    System.out.println(qury);
    //send and execute the query in database s/w. int res=0;
    if (res!=null)
        int res = st.executeUpdate(qury);

    //process results
    if (res==0)
        System.out.println("no record found");
}

```

```

else
    System.out.println(msg + "no. of records are updated");

} // try

catch (ClassNotFoundException cnf) // handles known exception
{
    cnf.printStackTrace();
}

catch (SQLException se) // handles known exception
{
    se.printStackTrace();
}

catch (Exception e) // handles unknown exception
{
    e.printStackTrace();
}

finally
{
    // close jdbc objects

    try
    {
        if (st != null)
            st.close();
    }

    catch (SQLException se)
    {
        se.printStackTrace();
    }

    try
    {
        if (con != null)
            con.close();
    }

    catch (SQLException se)
    {
        se.printStackTrace();
    }
} // finally

} // main

} // class

```

- \* In java local variables do not have initial values/default values so when they are declared they must be assigned with initial values based on their data types.
- \* create table, drop table, alter table etc queries are called as DDL queries. In real time programmers are not responsible to do these DDL operations through queries execution.
- \* Every Company contains DBT (database team) and that team is responsible to design the tables by satisfying all the six forms of normalization rules.

### NetBeans :

type : IDE for java environment

Version : 6.7.1 (compatible with jdk 1.6)

Vendor : sun microsystems (oracle corporation)

open source software (free software)

To download software : [www.netbeans.org](http://www.netbeans.org)

for help : [www.netbeans.org](http://www.netbeans.org)

Procedure to develop java application by using netbeans IDE to drop database table

Step(1) : Launch netbeans IDE

Step(2) : create java project from netbeans IDE

File menu → new Project → java → Java application → next →  
 Project Name → deselect create main class → select main project  
 → finish.

Step(3) : Add java class to the project

Right click on the project → new → Java class → class name  
 (drop test) → finish

PSVtn + tab space

For various short cuts of netbeans IDE refer help section keyboard  
 keyboard shortcuts.

ctrl+shift+i → package import

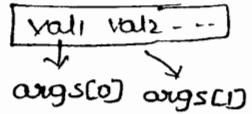
```
//DropTest.java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.Scanner;
public class DropTest
{
    public static void main(String args[])
        throws Exception
    {
        //read input values from keyboard
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter table name to drop");
        String tablename = sc.nextLine();
        //create JDBC connection object
        Class.forName("java.sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                                    "scott", "tiger");
        //create statement object
        Statement st = con.createStatement();
        //send and execute DDL query to db q/w
        int res = st.executeUpdate("drop table "+tablename);
        //process the results
        if (res == 0)
            System.out.println(tablename + " is not found");
        else
            System.out.println("Table dropped");
        //close JDBC objects
        st.close();
        con.close();
    } //main
} //class
```

Step(4): Run the application

Right click in the source code of droptest.java → run file → click on it

Procedure to set Command line argument values to java application of netbeans IDE project

Rightclick on project → run → main class [droptest] → arguments



How can we use single method call to execute both select and non-select queries?

(A) Use execute() method call on JDBC Statement object for this operation. This method call returns true when select query is executed and this method call returns false when non-select query is executed.

Prototype: public boolean execute (String query) throws SQLException.

This method call makes the programmer to gather the query execution results separately through other method calls.

if select query is executed (return value is true)

Resultset rs = st. getResultSet(); → gathers the result

if non-select query is executed (return value is false)

int cnt = st. getUpdateCount();

↓  
numeric value representing the no. of records that are effected due to non-select sql query execution.

\* working with execute() method is not recommended process

∴ use executeQuery() method for executing select query and

use executeUpdate() method for executing non-select query.

### Example on execute method

```
// create statement object  
Statement st = con.createStatement();  
  
// send and execute query  
// boolean flag = st.execute("select * from student");  
boolean flag = st.execute ("delete from student where sno=100");  
  
if (flag==true)  
{  
    System.out.println ("select query executed");  
    ResultSet rs= st.getResultSet();  
    Statement st1= st.executeQuery ("Select * from student");  
    while (rs.next ())  
    {  
        System.out.println (rs.getInt (1) + " " + rs.getString (2) + "  
                           " + rs.getString (3));  
    }  
    rs.close();  
}  
  
else  
{  
    System.out.println ("non-select query executed");  
    int cnt = st.getUpdateCount();  
    System.out.println ("no. of records that are effected " + cnt);  
}
```

\* Any vendor can develop JDBC driver for any database software by using one of the following four methodologies or mechanisms. They are

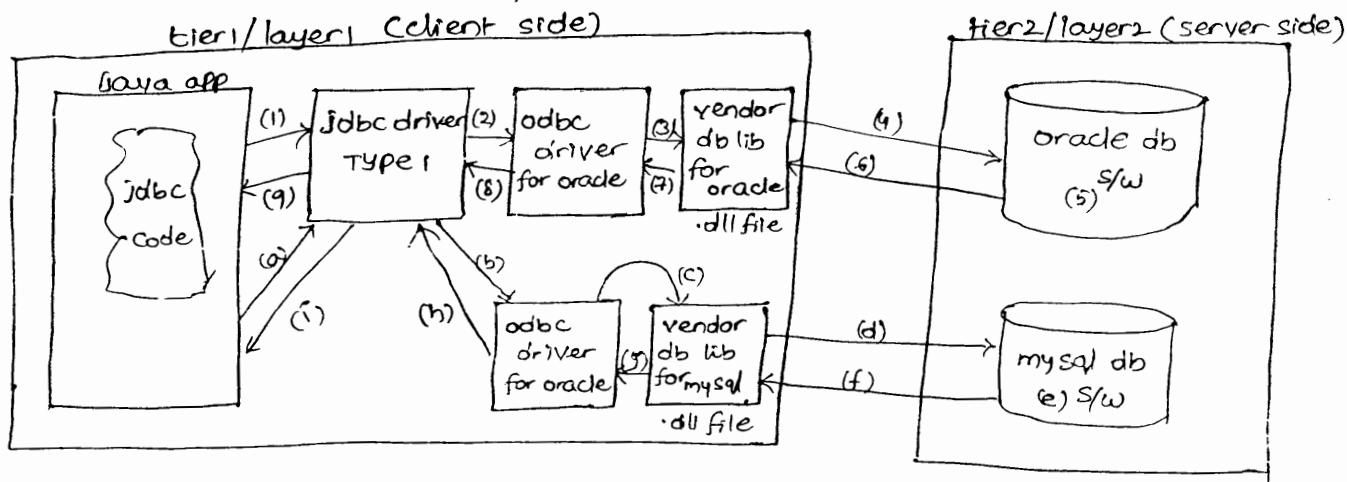
- (1) Type - 1 (Jdbc - Odbc bridge)
- (2) Type - 2 (Native - API / partly java driver)
- (3) Type - 3 (Net protocol / All java driver)
- (4) Type - 4 (Native - protocol / all - java driver)

\* Every database software gives one library called vendor db library having code and logic to locate and interact with db software. JDBC drivers and ODBC drivers take the support of these vendor db libraries to locate and communicate with their respective database softwares.

\* Generally this vendor database library comes as .dll file. In case of oracle this file name is oci.dll (comes with oracle s/w installation).

\* OCI - Oracle Connection Interface.

### Type 1 mechanism / methodology / Architecture



\* In client server environment in order to work with centralized oracle database software the server oracle software will be installed on the server machine of the network and client oracle software will be installed in the remaining machines of the network. Both client and server oracle software installations give vendor db library for oracle.

### Advantages:

- \* Type-1 driver is not specific to any database software. It uses an appropriate database software specific ODBC driver to interact with database softwares. Only SunMicro Systems is supplying JDBC driver based on type-1 mechanism.
- \* Can be used to interact with any db softwares.

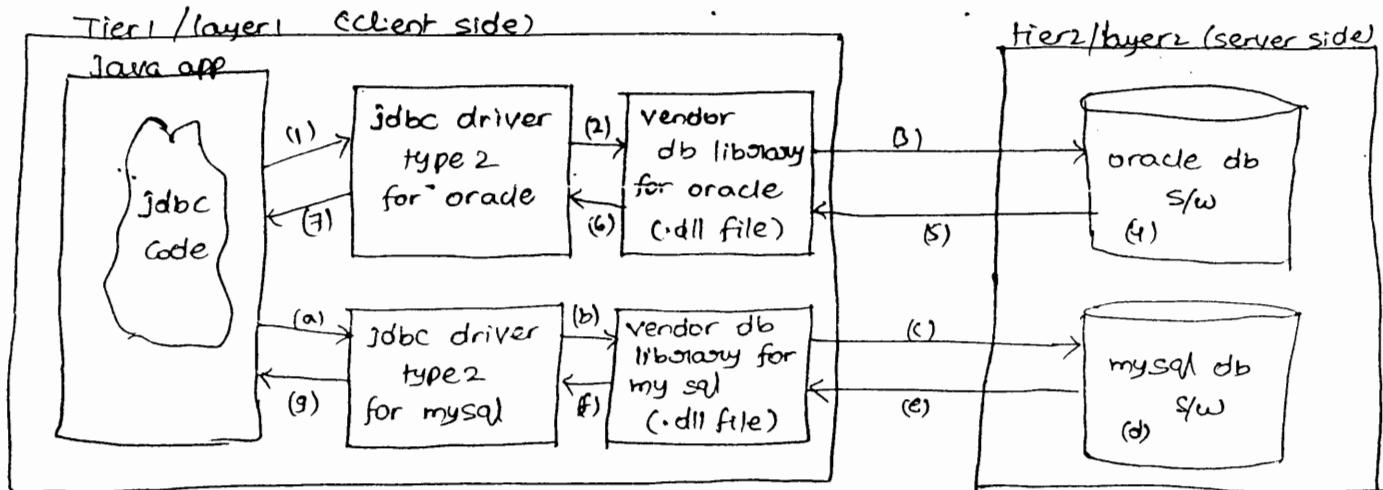
- \* Comes as built-in driver of SDK software so no need of arranging this driver separately.
- \* Since ODBC drivers are available almost for all database software we can use type-1 driver to interact with all the database softwares.

### Disadvantages

- \* Because of multiple conversions the performance is poor.
- \* Since vendor db library and ODBC driver are required at client side
  - Type-1 driver is not suitable for internet based applications.
  - Type-1 driver is not suitable for untrusted applet to database software communication.
- \* Type-1 driver is not suitable in large scale projects.
- \* Trusted applet can interact with files on filesystem to perform read and write operations whereas untrusted applets can not do this work.

### Type-2 mechanism:

TYPE2 mechanism/ methodology/ Architecture



- \* Type-2 JDBC driver uses native code to communicate with vendor db library. Except type-1 mechanism no other mechanism takes the support of ODBC drivers.
- \* Type-2 JDBC driver takes the support of db vendor db library directly to interact with database software.

## Advantages

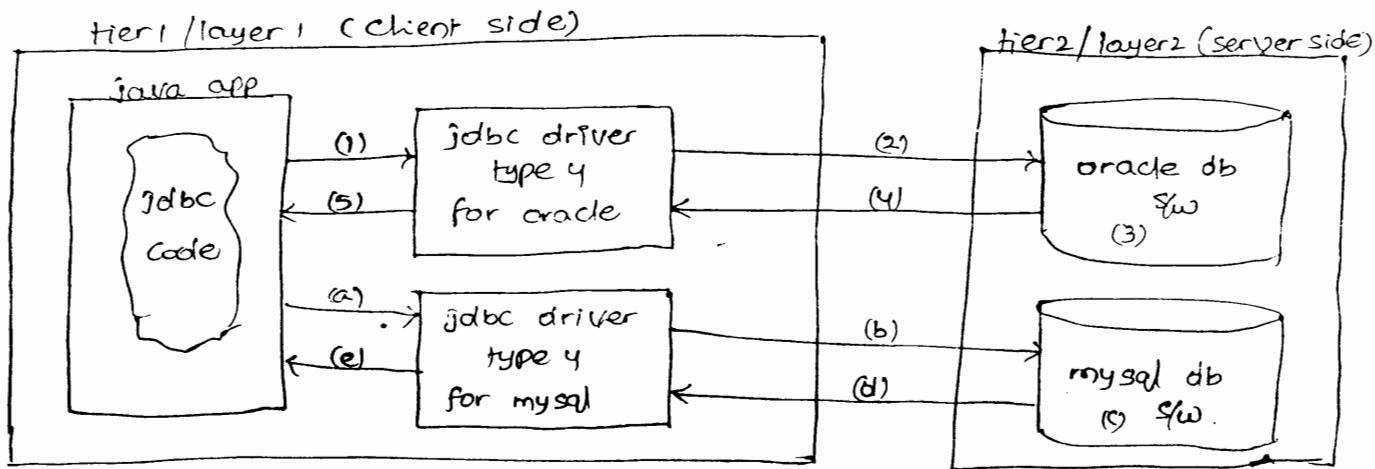
- \* No need of working with odbc drivers.
- \* use significantly better performance compare to type1 driver.

## disadvantages

- \* For every database software we need one <sup>seperate</sup> jdbc driver type-2.
- \* since jdbc driver & vendor db library is required at client side we can not use type-2 driver in internet based applications and for untrusted applet to database software communication.
- \* Not suitable to large scale applications.
- \* programmer needs to arrange type-2 jdbc drivers separately

## Type-4 mechanism:

TYPE 4 mechanism/ methodology/ Architecture



- \* Type-4 driver is designed to interact with database softwares directly without taking support of vendor db library and odbc driver.

## Advantages

- \* gives better performance compare to type-1 and type-2 drivers.
- \* NO need of working with odbc drivers and vendor db libraries.
- \* This driver will be developed purely in java environment so it exhibits platform independence and environment neutrality.

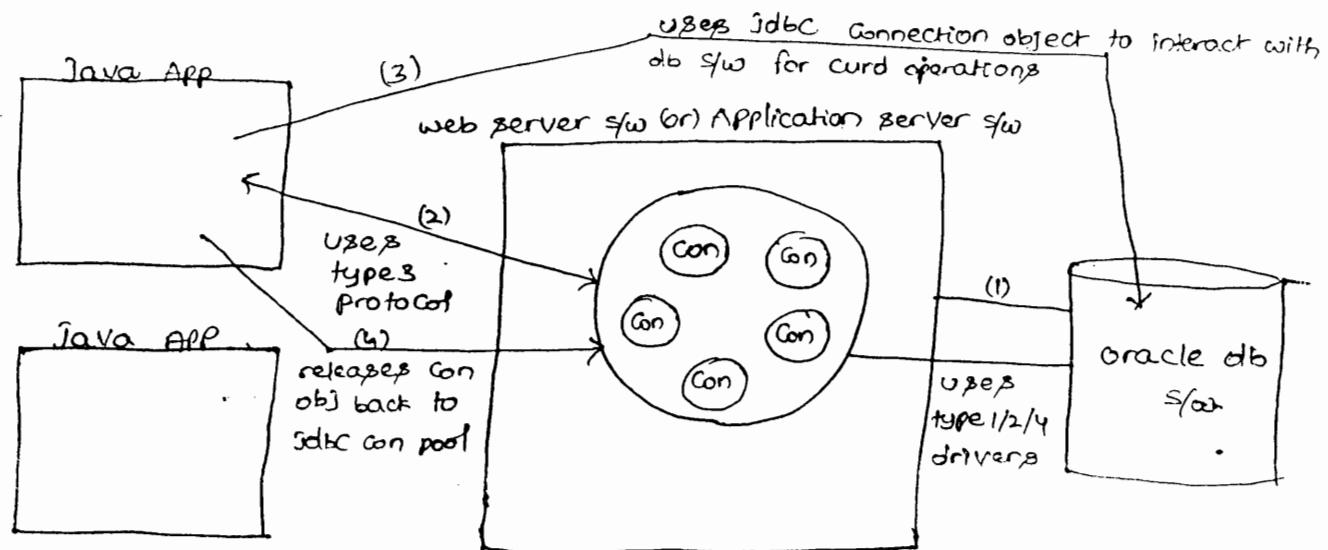
- \* This driver can be downloaded to client machine dynamically from internet network.
- \* suitable for internet programming based applications and for untrusted applet to database software communication.
- \* suitable for large scale projects.

### Disadvantages

- \* For every database software we need one jdbc driver type-4.
- \* Programmer must arrange jdbc drivers separately because they don't come along with jdk software.

### Type 3 mechanism

- \* Type 3 is not real jdbc driver if is given as protocol having set of rules to get communication between java application and jdbc connection pool.
- \* Jdbc connection pool is a factory that contains set of readyly available jdbc connection objects before actually being used.
- \* The advantage of jdbc connection pooling is we can use less no. of jdbc connection objects to make more java applications interacting with database softwares.
- \* Jdbc connection pools will be created and managed by web server, application server softwares like tomcat, weblogic and etc.



\* All jdbc connection objects of jdbc connection pool represents connectivity with same and single database software.

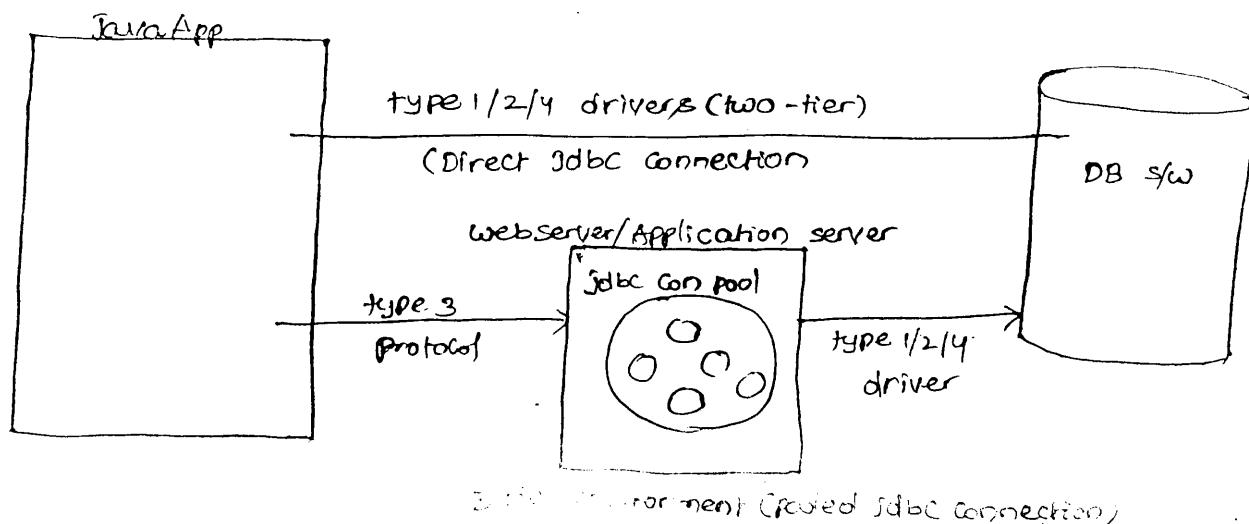
Ex: JDBC Connection pool for Oracle means all JDBC connection objects in that pool represents connectivity with same database software (Oracle).

With respect to diagram

- (1) Web server or application server uses type 1 (or) type 2 (or) type 4 driver to interact with database software and to create connection objects in connection pool.
- (2) Java application uses type 3 protocol to get one connection object from connection pool.
- (3) Java application uses that connection object to create other JDBC objects and to perform CURD operations on database.
- (4) Java application releases the connection object back to connection pool once task is completed. This connection object now becomes free to give service to other clients.

\* While working with JDBC connection pool the programmer is not responsible to create, manage and destroy JDBC connection objects from Java application. All these operations will be taken care by JDBC connection pool.

Summary diagram of JDBC drivers (all)



\*The JDBC connection that is created by the programmer directly and manually is called as direct JDBC connection object.

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

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

Direct JDBC Connection object.

\* The JDBC connection object that is collected from JDBC Connection pool is called as pooled connection object.

↔ For more information about JDBC drivers and architecture refer page nos 10-17 of booklet.

\* Type 3 is given to work with JDBC Connection Pool environment.

what is the JDBC driver that you have utilized in your project?

(A) If project is small scale application like two-tier application or office automation application then use JDBC driver type-4. If project is medium scale or large scale project like three-tier application like web application, bank application and etc.. then use type 3 with type-4 JDBC drivers. Here type-4 driver will be used to create connection objects in connection and type-3 protocol will be used to get connection objects from connection pool.

what is the difference between path and class path?

(B) Path: In order to make .cmd, .bat and .exe files of one directory executable from any location of computer then add the address location of these files to path environment variable.

\* .bat file combines set of DOS Commands or Windows Commands into single unit to execute them in sequence.

Problem:

```
e:\abc\run.bat
```

```
-----  
date  
time  
dir
```

e:\abc> run (success)  
e:\abc> run.bat (success)  
e:\> run (fails)  
run is not an external or internal command  
d:\demo> run (fails)  
run is not an internal or external command.

Solution: work with path environment variables

Take the address location of run.bat file i.e., e:\abc folder and add that to path environment variable.

mycomputer → Properties → Advanced tab → env variables →

variable name : **PATH**

value : **e:\abc; <other existing values>;.**

→ok →ok →ok

e:\abc> run (success)

e:\> run (success)

d:\demo> run (success)

c:\> run (success)

\* All our java tools like javac, java, javap are given as .exe files in java-Home\bin directory so to make them executing from any folder of your computer then add java-Home\bin directory to path environment variable.

\* Don't set environment variable values (like path and classpath) from command prompt. Always use myComputer properties related environment variables.

\* Multiple values added to environment variable must be separated with ";" symbol. we should not use this ";" symbol as beginner of the first value as the terminator of last value.

\* when new values are added to my computer environment variable they will be reflected only in the new command prompts that are opened after adding new values.

### Class Path:

\* when Java application uses new APIs and/or third party APIs (other than JDK APIs) then the new APIs, third parties APIs related directories are JAR files must be added to class path environment variable to make Java applications, Java tools recognizing that new API, third party API.

\* classpath is purely related to Java environment.

\* API in Java language is nothing but set of classes and interfaces, which come in form of packages.

### Understanding Problem:

E:\Demo\ABC

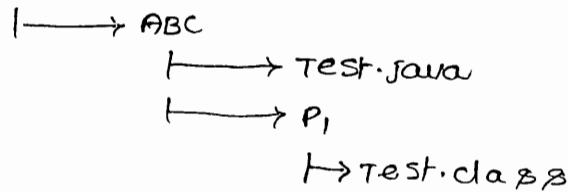
Test.java

```
Package P1;  
Public class Test  
{  
    Public void m1()  
    {  
        System.out.println("Test:m1()");  
    }  
}
```

e:\Demo\ABC > javac -d . Test.java

gives Test.class in P1 package.

E:\Demo



D:\work  
|→ mainApp.java

### MainApp.java

```
import P1.Test;  
public class MainApp  
{  
    public static void main (String args [])  
    {  
        Test t = new Test();  
        t.m1();  
    } // main  
} // class
```

D:\work> javac MainApp.java (x → failure)  
error: can not find symbols P1, Test, m1()

Applying solution: work with class path

Add the directory location where "P1" package is available (e:\demo\ABC) to CLASSPATH environment variables variable.

My Computer → Properties → Advanced tab → Env Variables → New →

Variable name: CLASSPATH  
Value : e:\demo\ABC;  
→ OK → OK → OK

(or)

My Computer → Properties → Advanced tab → Env Variables → edit →

Variable name: CLASSPATH  
Value : e:\demo\ABC; <other existing values>;  
→ OK → OK → OK

D:\work> javac MainApp.java (success)  
gives MainApp.class

D:\work> java MainApp  
→ gives output.

- \* To make java tools running anywhere in your computer then work with Path environment variable. To make these java tools to recognizing and utilizing third party APIs and new APIs then work with CLASSPATH environment variable.
- \* Generally programmers and third party vendors keep their APIs in jar files and they will supply them to others. These other programmers use these new APIs and third party APIs by adding their jar files to classpath.

E:\Demo\ABC> jar cf sathyajar.jar .

C → create jar file  
f → use given name as jar file name.

- \* To add jar file to classpath

myComputer → properties → Advanced tab → env variables → edit →

variable name: CLASSPATH

value : e:\Demo\ABC\sathyajar.jar; <other existing values>;

→ OK → OK → OK

- \* Oracle corporation supplies its type 4 mechanism, type 2 mechanism related JDBC drivers in the form of "ojdbc14.jar" file along with oracle software installation.

- \* The location of ojdbc14.jar file is

oracle\_home\product\10.2.0\db\_1\jdbc\lib (oracle 10g)

\* Jar file can represent new APIs or third party APIs.

\* Jar file can represent JDBC drivers.

\* Jar file can represent a package webapplication (web site).

\* Jar file can represent an EJB component.

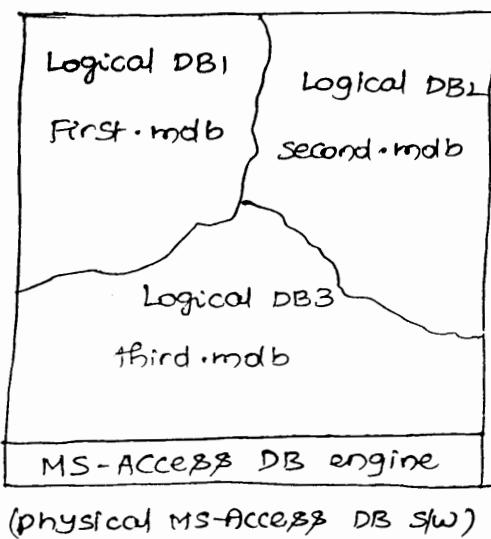
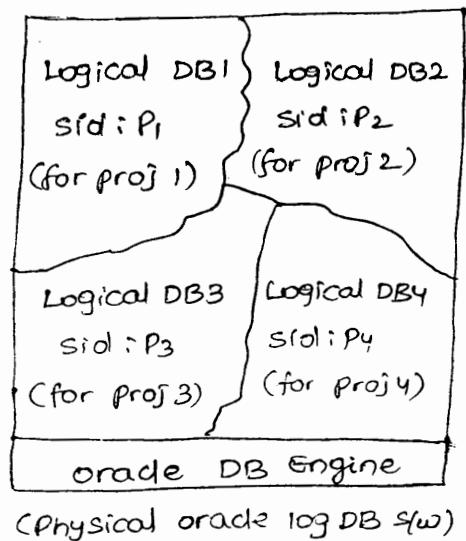
\* Jar file can combine multiple resources into single file like zip file.

\* In every physically installed database software we can create multiple logical partitions as logical database.

\* Every logical database will be identified with its database name.

\* In oracle database software database name or logical database is called as sid.

Ex: If multiple projects of a company are utilizing same oracle database software then the oracle database software should be installed only once in common machine of a company and multiple logical databases will be created in that oracle database software for multiple projects on one per project basis.



\* In MS-Access each .mdb file is one logical database and it will be identified with its file name.

\* Creating new logical databases in physical database software is the responsibility of DBA.

\* In each logical database multiple DB tables, DB users and etc. can be there.

\* When Oracle software is installed it will give one default logical database, to know sid of this logical database:

Approach(1):

```
SQL> select * from GLOBAL_NAME;
```

ORCL> REGEXP\_LIKE . . . . .

→ sid of current logical database (default).

approach (2): open oracle-home\product\10.2.0\db-1\NETWORK\ADMIN\tnsnames.ora

file and look for service-name = orcl entry.

↳ sid of logical database.

approach (3):

Control panel → performance and maintenance → administrative tools →

services → Look for oracle service ORCL

↳ sid of logical database.

\* If logical database of physical database software is allowing multiple users or client applications concurrently to manipulate the data then it is called as multiuser database software. If only one user (or) client application is allowed to perform the above said operation then it is called single user database software.

Examples of multiuser database softwares

oracle, sybase, mysql, Postgresql and etc...

Examples of singleuser database softwares

MS-Access, foxpro, foxbase and etc...

Working with jar Command:

E:\temp

    → unit1

        → a.txt

        → b.txt

to create jar file

e:\temp\unit1> jar cf sathya.jar .

C → create jar file

f → with specified name as jar file name.

to see the table of Content of jar file

e:\temp\unit1> jar tf sathya.jar

t → Table of Content in jar file.

\* Every jar file contains one manifest.mf file in META-INF folder.

This manifest file contains description about various resources of jar file.

to add new file to existing jar file

e:\temp

| → unit

  | → c.txt (create new file)

e:\temp\unit> jar uf sathya.jar c.txt

Adds c.txt file to sathya.jar file.

v → updates the existing jar file

to extract files from sathya.jar file

to extract specific file

e:\temp\unit> jar xf sathya.jar a.txt

to extract all files from jar file

e:\temp\unit> jar xf sathya.jar

x → Extraction of jar file

\* we can view and extract the content of jar file either by using winzip tool or winrar tool.

\* The oracle corporation supplied type 2 mechanism based JDBC driver for oracle is called as "oracle oci" driver.

\* The oracle corporation supplied type 4 mechanism based JDBC driver for oracle is called as "oracle thin" driver.

\* Every software installed in the computer resides in a logical position called software port and every software port will be identified with its port number.

\* Total 65535 software ports can be there in a single computer. In that 1 to 1024 are reserved for operating system services.

oracle software → default port no: 1521

Tomcat software → default port no: 8080

weblogic software → default port no: 7001

Internet Explorer → default port no: 1078

\* socket no is: (IP address / host name of machine) + (port no of software

\* port no. of oracle software is: 1521 service)

\* socket no. of oracle db software when installed on a machine having IP address 180.45.67.4

180.45.67.4 + 1521

socket no. of oracle db s/w.

\* To communicate with any software installed on any computer we must know its socket number.

Details of oracle corporation supplied type 2 mechanism based jdbc driver for oracle

Vendor: oracle corporation

Commercial jdbc driver

mechanism: type 2 mechanism

technical name: oracle oci driver

↓  
oracle connection interface

jdbc driver class: oracle.jdbc.driver.OracleDriver

↓  
package name      ↓  
class implementing java.sql.Driver  
Interface

Database url: jdbc:oracle:oci8:@<sid>

↓      ↓      ↓  
Protocol      Subname      logical database name

Jar file representing this oracle oci driver

oracle 8i → classes11.jar or classes12.jar

oracle 9i → classes11i.jar or classes12.jar or ojdbc14.jar

oracle 10g → classes12.jar or ojdbc14.jar

oracle 11g → ojdbc6.jar

## Procedure to develop java application by using oracle oci driver

Step(1): Gather details about oci driver as shown above

Step(2): Develop java application by using oracle oci driver as shown below

```
//selectTest.java
```

```
import java.sql.*;
import java.util.*;
public class Test
{
    public static void main (String args[])
        throws Exception
    {
        //load jdbc driver class
        Class.forName ("oracle.jdbc.driver.OracleDriver");
        //establish connection with DB s/w
        Connection Con = DriverManager.getConnection ("jdbc:oracle:oci8:",
                                                    "@orcl*", "scott", "tiger");
        //create jdbc statement object
        Statement st = Con.createStatement();
        //send and execute sql query in db s/w and gather the values
        ResultSet rs = st.executeQuery ("select * from student");
        //process the result-set object
        while (rs.next())
        {
            System.out.println (rs.getInt(1) + " " + rs.getString(2) + " "
                               + rs.getString(3));
        }
        //close jdbc objects
        rs.close();
        st.close();
        con.close();
    }
}

// javac selectTest.java
// java selectTest
```

Step(3): Add ojdbc14.jar file or other related equivalent jar files to class path environment variable.

my Computer → properties → advanced tab → env variables → edit

variable name: CLASSPATH

value : oracle\product\10.2.0\db\_1\jdbc\lib\ojdbc14.jar;  
<other existing values>;

→ok →ok →ok

Step(4): Compile and execute the application.

>javac selectTest.java

>java selectTest

\* we can write jdbc code and we can create jdbc objects like statement object, resultSet object and etc in a common way irrespective of the jdbc driver that we are using.

\* As of now sun micro system is not supplying type2 and type4 mechanism based drivers so the programmer must gather them from database vendors or third party vendors.

Oracle corporation supplied type4 mechanism based jdbc driver for oracle

Technical name: oracle thin driver

mechanism : type 4

Vendor : oracle corp

Commercial jdbc driver

Jdbc driver class : oracle.jdbc.driver.OracleDriver  
(or)  
oracle.jdbc.OracleDriver

Jdbc url /db url : jdbc:oracle:thin:@<hostname/IP address>:<port no>  
↓                   ↓  
Protocol      Subname

↓  
of computer where : <sid> of oracle db s/w  
oracle db s/w is installed logical db (1521)

JAR files that represents this driver : same as oracle oci driver

why the db url of type 4 jdbc drivers is quite lengthy when compare to the db url of type 1, type 2 drivers?

(A) Type 1 driver is designed to talk with database software through odbc driver so its db url just need to locate odbc driver (jdbc:odbc:<dsn>)

once type 2 driver locates vendor db library of client side that db library takes care of communicating with database software so its db url just need to locate vendor db library (jdbc:oracle:oci8:@<sid>).

Type 4 drivers needs to locate and communicate with database softwares directly without taking support of other resources of client side, so its url contains multiple details (jdbc:oracle:thin:@<ipaddress>/hostname>: <portno>: <sid>).

Procedure to work with oracle thin driver in our java applications.

NOTE: To refer a computer being from current machine we use the word called local host.

Step(1): Gather the details of oracle thin driver as shown above.

Step(2): Develop java application by using oracle thin driver.

//selectTest.java

same as previous class application but the following db url to create jdbc connection object.

//establish connection with DB sys

```
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:  
1521:orcl","scott","tiger");
```

Step(3): Addojdbc14.jar file or other equivalent jar files to class path.  
same as previous class.

Step(4): Compile and execute the java application.

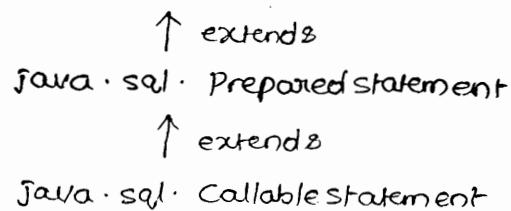
```
>javac selectTest.java
```

```
>java selectTest
```

\* The values, jar files added to my Computer environment variable class path are not visible to the Projects created in IDE softwares. That means we need to add jar files to the libraries of IDE project separately.

\* If java application of the netbeans IDE project uses oracle thin driver then the oracle thin driver related ojdbc14.jar must be added to the library folder of the project (Right click on libraries folder → add jar → Browse and select ojdbc14.jar).

### java.sql.Statement

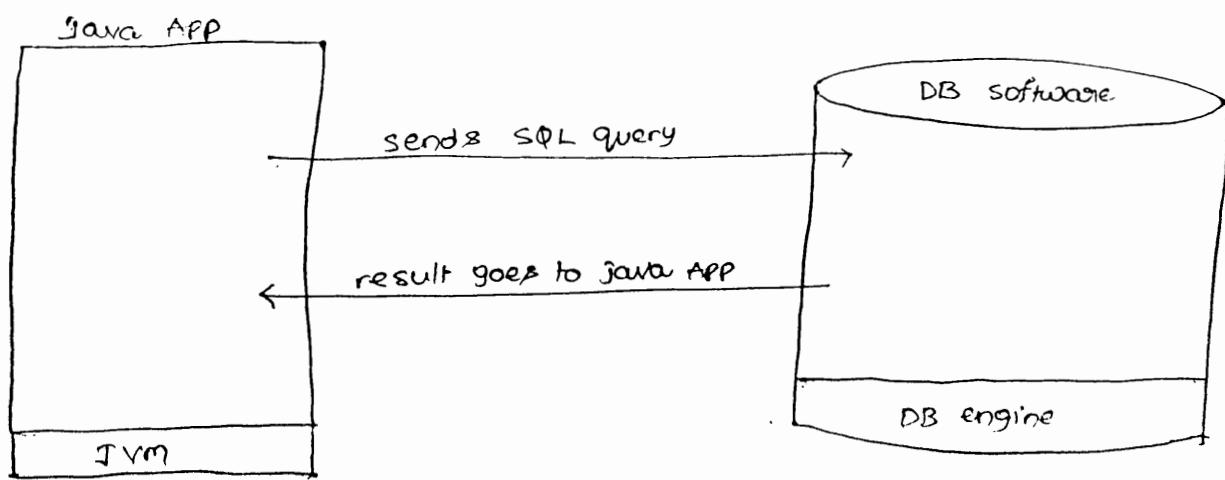


\* There are 3 types of JDBC statement objects which can be used in our java application to send and execute sql queries in database software.

(1) Simple statement object (object of a java class that implements `java.sql.Statement` interface)

(2) Prepared Statement object (object of a java class that implements `java.sql.PreparedStatement` interface)

(3) Callable Statement object (object of a <sup>java</sup> class that implements `java.sql.CallableStatement` interface)



\* The database engine of database software performs following operations on SQL query that comes from client application.

(1) Parse (2) Execute (3) Fetch.

\* In parse operation the syntax of the query will be verified by splitting the query into multiple tokens.

\* In execute operation the parsed query will be executed and its output will be stored in the buffer.

\* In fetch operation results from the buffer will be gathered and results will be delivered to client application.

#### Limitations with simple statement object :

\* Framing SQL query with variable for simple statement object is complex process.

\* If simple statement is used to execute same query for multiple time with same or different values then

a) same query goes to database software for multiple times.

b) same query will be parsed in database software for multiple times.

c) same query will be executed in database software for multiple times with same (or) different values.

d) The same query output will be fetched out for multiple times.

\* In that above operations (a) and (b) are unnecessary operations done for multiple times on single query. But they can not be avoided while working with simple statement object. This makes database software to perform unnecessary operations which may degrade the performance of the application.

\* To overcome above limitations use precompiled SQL query with the support of prepared statement object.

Note: The limitation (2) talks about the simple statement can't execute the query multiple times in db SW by just sending and parsing that query only for one time in database software.

\* Simple Statement object sends raw SQL query from Java application to database software every time so the query goes through parsing, execution and fetch operations on every query each time.

\* The jdbc Prepared statement object deals with pre Compiled sql query.

\* The query that comes to database software from java application only once, becomes parsed query only once and allows client application (java application) to set the values for the query for multiple times, to execute the query for multiple times, time with same or different values to gather the results for multiple times is called as precompiled sql query.

\* JDBC prepared statement object resides in java application and represent pre compiled sql query of database software so the programmer can provide further instructions to precompiled query like assigning values, query execution and etc... by using this prepared statement object from java application.

\* To execute SQL query only for one time in database software use simple statement object. Similarly to execute same SQL query for multiple times in database software with different or same values use JDBC prepared statement object.

Procedure to work with jdbc prepared statement object

(1) Prepare SQL query with parameters / place holders (?)

```
String query = "insert into student values(?, ?, ?);  
                  ^  ^  ^";
```

### Placeholders

place holders/parameters

(2) Send SQL query to database software as pre-compiled query and represent that query with Prepared Statement object.

Prepared statement ps = Con. Preparestatement (query);

This object represents

**Prep** PreCompiled query of db sw being from java application.

This method sends given sql query to db s/w and makes that query as PreCompiled query in db s/w.

(3) Set values to parameters of precompiled query by using setXXX() methods.

```
ps.setInt(1, 567);
ps.setString(2, "Raja");
ps.setString(3, "hyd");
```

NOTE: The parameters of SQL query will have one based index.

Here "hyd" is parameter value.

(4) Execute the query in database software.

```
int result = ps.executeUpdate();
```

(5) Process the results

```
if (result == 0)
    System.out.println("Record insertion is failed");
else
    System.out.println("Record insertion is successful");
```

(6) To execute same SQL query for multiple times with same or different values repeat steps 3 to steps for multiple times.

(7) Close JDBC Prepared Statement object.

```
ps.close();
```

Write a Java application to read multiple student details from keyboard and insert them to database table as records.

NOTE: Here we need to execute same SQL query for multiple times for that works with precompiled SQL query with support of PreparedStatement object.

```
// PstTest.java (with coding standards)
// Insert multiple student details to DB table
// version: 1.0
// author: team-s
import java.sql.*; import java.util.*;
public class PstTest
{
    public static void main (String args[])
    {
```

```

Connection con=null;
PreparedStatement ps=null;
Scanner sc=null;
sc=new Scanner(System.in);
int n=0;
if(sc!=null)
{
    //read student count from end user
    System.out.println("Enter no. of student");
    n=sc.nextInt();
}
//if
try{
    //write jdbc code
    //create jdbc connection object
    Class.forName("oracle.jdbc.driver.OracleDriver");
    con=DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                    :1521:", "scott", "tiger");
                                    ↓
                                    orcl
    //prepare sql query with parameters
    String query="insert into student values(?, ?, ?)";
    //create prepared statement object representing precompiled
    //sql query
    if(con!=null)
    {
        ps=con.prepareStatement(query);
    }
    //read each student details from keyboard and insert them
    //to database table as record.
    if(sc!=null & ps!=null)
    {
        for(int i=1; i<=n; i++)
        {
            System.out.println("Enter "+i+" student details");
            System.out.println("Enter no:");
            int no=sc.nextInt();
            System.out.println("Enter name:");
            int newString name=sc.next();
        }
    }
}

```

```
    system.out.println ("Enter address:");
    String address = sc.next();

    //set the above values to parameters of query
    ps.setInt (1, no);
    ps.setString (2, name);
    ps.setString (3, address);

    //execute the query
    int steps = ps.executeUpdate();

    //process the result
    if (steps == 0)
    {
        system.out.println (i + " student details are not inserted");
    }
    else
    {
        system.out.println (i + " student details are inserted");
    }

} //for
} //if
} //try
catch ( ClassNotFoundException cnf )
{
    cnf.printStackTrace();
}
catch ( SQLException se )
{
    se.printStackTrace();
}
catch ( Exception e )
{
    e.printStackTrace();
}

finally
{
    try
    {
        if (ps != null)
            ps.close();
    }
    catch ( SQLException se )
    {
        se.printStackTrace();
    }
}
```

```

try {
    if (con != null)
        con.close();
}
catch (SQLException sel) {
    sel.printStackTrace();
}
// finally
// main
// class
//> javac PstTest.java
//> java PstTest

* Compare to simple statement object the jdbc prepared statement object
reduces network traffic between java application and database soft-
ware in each round trip.

* The setxxx(-) methods allow the programmer to set values to sql
query or query parameters in java notation so working with string
type input values for sql query becomes easy while working with
jdbc prepared statement object.

* By using prepared statement object we can execute both select and non
select sql queries with zero or more parameters.

* Provide one based index sql query parameters from left to right order.

Example to execute select sql query by using jdbc PreparedStatement
object

// create PreparedStatement object
PreparedStatement ps = con.prepareStatement("select * from student
where sno >=? and sno <=?");
// set values to the parameters of query
ps.setInt(1, 100);
ps.setInt(2, 300);

```

```

    //execute query
    ResultSet rs = ps.executeQuery();
    //process results
    while(rs.next())
    {
        System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
    }

```

\* The SQL query of JDBC PreparedStatement object can be there with or without parameters.

```

//create JDBC PreparedStatement object
PreparedStatement ps = con.prepareStatement("select * from student");
//execute query
ResultSet rs = ps.executeQuery();
//process results
while(rs.next())
{
    System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
}

```

query without parameters.

\* The parameters of PreparedStatement object related SQL query can represent only condition values and input values but they can not represent table names, column name, or the SQL keywords of the query.

select \* from studen where sname like ?      (valid query)

select \* from ? where ? like ?      (invalid query)

select \* ? student where sname like ?      (invalid query)

select \* from Student where sno >= ? and sno <= 200      (valid query)

\* One Java application can have multiple JDBC Connection objects, Statement objects and ResultSet objects.

\* Generally Java JDBC applications are two tier applications containing three layers.

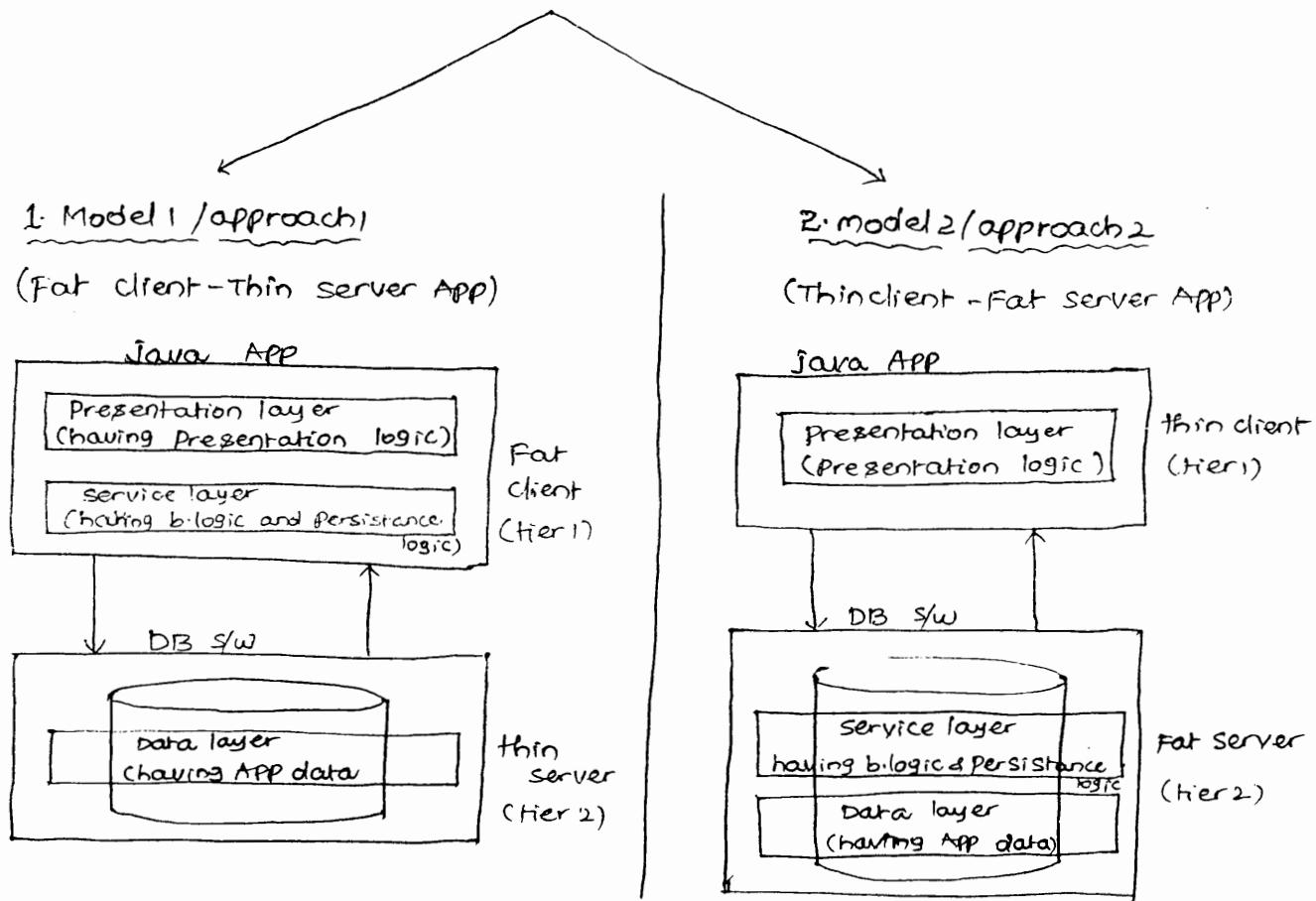
(1) Presentation layer having presentation logic

(2) Service layer having business logic and persistence logic.

(3) Data layer having application data.

\* Data layer is responsible to maintain, manage application data with the support of persistence stores like business logic, persistence logic and etc.

The two models of developing JDBC two-tier applications



\* The JDBC applications which deals with simple statement object, Prepared statement object are called as "Fat client - thin server" applications.

+ In thin client and Fat server applications the business and persistence logic will be placed in DB s/w as PL/SQL procedures and functions and to call them from java application the JDBC CallableStatement object will be utilized.

\* since PL/SQL Procedures / functions resides and execute permanently in the database software they are also called as stored procedures and functions.

\* PL/SQL procedure doesn't return a value whereas PL/SQL function returns a value.

\* The PL/SQL function, procedure related parameters contains mode along with type.

modes are

In (default)

out

Inout

\* If logic is

$y := x * x;$

then

$x \rightarrow$  In parameter

$y \rightarrow$  out parameter

$x := x * x;$

$x \rightarrow$  Inout parameter.

\* We can gather results from PL/SQL Procedure from out parameters or Inout parameters.

\* We can gather results from PL/SQL function either as return value or outparameters / Inout parameters.

\* PL/SQL programming, SQL queries are database software dependent. So

JDBC persistence logic is database software dependent persistence logic. To develop database software independent persistence logic use ORM software like Hibernate. (Object Relational mapping - ORM)

steps to develop PL/SQL Procedure in Oracle

Step(1): Launch SQL\*plus tool

Step(2): Execute any dummy query

Step(3): Launch ed editor

↳ SQL> ed

Step(4): Type and save PL/SQL Procedure.

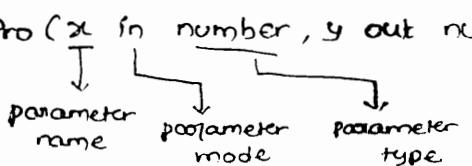
create or replace  $\downarrow$  first\_Pro ( $x$  in number,  $y$  out number)  
as  
procedure.

begin

$y := x * x;$

end;

/



Step(5): Execute the PL/SQL procedure

File menu → save → exit → ~~SQL>/~~

Procedure to work with jdbc callable statement object to call PL/SQL procedure of Database software

Step(1): prepare query calling PL/SQL procedure.

String query = "call first\_pro (?, ?)";  
↓  
Represents parameters

Step(2): create CallableStatement object having the above query.

CallableStatement cs = con.prepareCall(query);  
↓  
Represents the  
PLSQL procedure  
on the db s/w  
(first\_pro)

Step(3): set input values to in parameters of PL/SQL procedure call by using setXXX(-) methods.

cs.setInt(1, 20);  
↓      ↓  
parameter index      parameter value

Step(4): Register out parameters with jdbc data types.

cs.registerOutParameter(2, Types.INTEGER);  
↓  
parameter index      ↑  
                        jdbc type

\* Every out parameter must be registered with jdbc type. All jdbc data types are bridge datatypes between underlying database software datatypes and java data types. They help jdbc driver to convert database software notation result into java notation result while working with PL/SQL procedure or functions.

<u>Java data type</u>	<u>jdbc datatype</u>	<u>oracle data type</u>
int	TYPEs.INTEGER	number
long	TYPEs.LONG	number
String	TYPEs.VARCHAR	varchar2
byte[]	TYPEs.BINARY	BLOB
java.sql.Date	TYPEs.DATE	date
:	:	:

\* All jdbc datatypes are constants of java.sql.Types class.

\* Public static final member variables of a java class/ java interface are called as constants.

NOTE: IN parameters of PL/SQL procedures and functions need not to be register with jdbc types because the jdbc driver can detect their jdbc datatypes automatically based on the values that are set and based on the setXXY(-) methods that are used to set the values to IN parameters.

Step(5) : Make database software executing PL/SQL procedure.

```
cs.execute();
```

Step(6) : Gather results from OUT parameters of PL/SQL procedure.

```
int result = cs.getInt(2);
```

↓      ↴ out parameter index.  
decide this method  
based on the jdbc type  
that is used to register out parameter

Step(7) : To call PL/SQL Procedure for multiple times with same or different values then repeat step(3) to step(6) for multiple times (except step(4)).

Step(8) : close jdbc callable statement object.

```
cs.close();
```

\* If multiple applications of a module (or) multiple modules of a project (or) multiple projects of a company are looking to use same business logic or persistance logic then instead of writing those

logic in every application of a module (or) in every module of a project (or) in every project of a Company it is recommended to write only once as centralized logic in DB s/w in the form of PL/SQL procedure or function and use them in multiple applications of a module (or) in multiple modules of a project (or) in multiple projects of a Company having reusability.

#### Example application based on above steps

```
// CTest.java
import java.sql.*;
import java.util.*;
public class CTest
{
    public static void main(String args[])
        throws Exception
    {
        // read input values from key board
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a value");
        int no = sc.nextInt();
        // create jdbc connection object
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:",
            "localhost:1521:orcl",
            "scott", "tiger");
        // prepare sql query calling PL/SQL Procedure.
        String query = "{call first_pro(?, ?)}";
        // create callable statement object
        CallableStatement cs = con.prepareCall(query);
        // register OUT parameters with jdbc types.
        cs.registerOutParameter(2, Types.INTEGER);
        // set IN parameters
        cs.setInt(1, no);
```

```

//make db s/w to execute PL/SQL procedure
cs.execute();

//gather results from OUT parameters
int result = cs.getInt(2);
System.out.println("The Result is "+result);

//close JDBC objects
cs.close();
con.close();

}//main
}//class

//make sure that first-pro(-,-) PL/SQL procedure is available in
oracle DB s/w.

//>javac CTest.java
//>java CTest

```

\* Any mechanism based JDBC driver supplied by any vendor can work with all the 3 types of JDBC statement objects.

### PL/SQL Procedure in oracle dealing with SQL query

```

create or replace procedure emp-pro (eno in number, ename out varchar,
                                     bsal out number)

```

```

as
begin
select ename into name from emp where empno=eno;
select sal into bsal from emp where empno=eno;
           ↓          ↓
      column    output parameter
      name
end;

```

```

//CTest2.java
import java.sql.*;
import java.util.*;

public class CTest2
{
    public static void main (String args[])
        throws Exception
    {

```

```

    //read input values from keyboard
    scanner sc=new scanner (System.in);
    System.out.println ("Enter emp no:");
    int no = sc.nextInt();
    //create jdbc con obj
    Class.forName ("oracle.jdbc.driver.OracleDriver");
    Connection con= DriverManager.getConnection ("jdbc:oracle:thin:@localhost
                                              :1521:ord", "scott", "tiger");

    //Prepare query calling the PL/SQL procedure
    String query = "{ call emp_pro(?, ?, ?) }";
    //Create callable statement object
    CallableStatement cs = con.prepareCall (query);
    //register out parameters with jdbc types
    cs.registerOutParameter (2, Types.VARCHAR);
    cs.registerOutParameter (3, Types.INTEGER);
    //set values to IN parameters
    cs.setInt (1, no);
    //execute the PL/SQL procedure
    cs.execute();
    //Gather results from OUT parameters
    System.out.println ("emp name is:" + cs.getString (2));
    System.out.println ("emp sal is:" + cs.getInt (3));
    //close jdbc objects
    cs.close();
    con.close();
}

//main
} //class

```

- \* In PL/SQL programming of oracle we can use cursor datatype variables to store bunch of records given by SQL select query. For this we generally use sys\_ref cursor data type of oracle PL/SQL programming.
- \* cursor of PL/SQL programming is like jdbc resultSet object of java programming.

## Example application on cursor

```
create or replace procedure find_emps( fchar$ in varchar2, myrecords
                                     out sys_refcursor) as
begin
  open myrecords for
    select empno, ename, sal, job from emp where ename like fchar$;
end;
```

- \* Sunmicrosystem is not supplying jdbc type in Types class to register cursor type out parameters so use oracle corporation supplied OracleTypes.CURSOR constant as jdbc data type for the above said situation.
- \* To receive results from cursor type out parameters use getObject() method to receive jdbc resultset object.

```
//CursorCsTest.java
import java.sql.*;
import java.util.*;
import oracle.jdbc.*; //for oracle types class (this pkg is available in ojdbc14.jar).
class CursorCsTest
{
  public static void main (String args[]) throws Exception
  {
    //read cond character from keyboard
    Scanner sc = new Scanner (System.in);
    sop("Enter chars (first letters of emp name)");
    String cond = sc.next();
    cond = cond + "%"; //r becomes r%
    //write jdbc code
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                                :1521:ord", "scott", "tiger");
    // create callablestatement object
    CallableStatement cs = con.prepareCall("{call find_emps(?,?)}");
```

//register OUT parameters with JDBC types (oracle Corporation supplied  
JDBC types)

cs.registerOutParameter(2, OracleTypes.CURSOR);

//set values to IN parameter

cs.setString(1, cond);

//execute PL/SQL procedure

cs.execute();

//gather results from OUT parameter

ResultSet rs = (ResultSet) cs.getObject(2);

//display result

while(rs.next())

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

}

//close JDBC objects

rs.close();

cs.close();

con.close();

} //main()

} //class

\*The oracle PL/SQL programming provides a build built-in cursor variable called SQL%ROWCOUNT. This variable always holds no. of records that are effected when non select SQL queries are executed in PL/SQL Procedure or function.

Example application:

PL/SQL procedure in oracle (Based on non-select query)

create or replace Procedure fire\_emps (salrange in number, cnt out  
number) as

begin

delete from emp where sal > salrange;

cnt := SQL%ROWCOUNT;

end; ↳ returns a numeric value indicating no. of records that are effected because of above delete query execution

```

// Cstest3.java
public class Cstest3 {
    public static void main(String args[]) throws Exception {
        // read input values from keyboard
        Scanner sc = new Scanner(System.in)
        System.out.println("Enter salary range");
        int salrange = sc.nextInt();
        // create JDBC connection object
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:odbc:ocis@ord",
                                                    "scott", "tiger");
        // create CallableStatement object
        CallableStatement cs = con.prepareCall("{ call fire_emps(?,?) }");
        // register OUT parameters with JDBC types
        cs.registerOutParameter(2, Types.INTEGER);
        // set IN values
        cs.setInt(1, salrange);
        // execute PL/SQL Procedure
        cs.execute();
        // gather results from OUT parameters
        System.out.println("cnt of emps who are fired:" + cs.getInt(2));
        // close JDBC objects
        cs.close();
        con.close();
    } // main()
} // class

```

### Working with PL/SQL functions:

- \* PL/SQL function returns a value.
- \* Results from PL/SQL function can be gathered from either from out parameters or from return value. (we can even use both).
- \* While calling PL/SQL function the query must have a parameter representing return value and that parameter should be treated as out parameter

and should also be registered with JDBC type.

Example PL/SQL function:

create or replace function first\_fx(x in number, y out number) return  
number

28

3 number);

begin

$y := x * x;$

$\exists := x * x \neq x;$

return 3;

end;

Procedure to call PL/SQL function from java application by using jdbc

Callable Statement object

i) Prepare query calling PL/SQL function

String query = " { ? = call first-fx (? , ?) } ";

## Parameter 10

hold return value

## of PL/SQL function

2) create CallableStatement object representing PL/SQL function.

Callable Statement cs = con.prepareCall("query");

3) register return parameter, out parameter with jdbc types

```
cs.registerOutParameter(1, Types.INTEGER); //return parameter
```

cs.registerOutParameter(3, Type.B. INTEGER);

4) Set values in parameters

cs. setInt (2, 100);  
↓      ↓  
parameter   value  
index

5) Make database software executing PL/SQL function

cs.execute();

6) Gather results from return parameter, out parameters

```
int age1 = cs.getInt(1); // gives value 3;
```

```
int guess2 = cs.getInt(3); // gives square value (y);
```

S.O.P(PIE81+ " " +PIE82);

- 7) To call and execute PL/SQL function for multiple times with same or different input values repeat step(4) to step(6) for multiple times.  
8) close CallableStatement object

```
cs.close();
```

```
// CSFxTest.java
```

```
import java.sql.*;
import java.util.*;
public class CSFxTest
{
    public static void main (String args[]) throws Exception
    {
        //Read input values from key board
        Scanner sc = new Scanner (System.in);
        System.out.println("Enter value");
        int no = sc.nextInt();

        //Create JDBC Connection object
        Class.forName ("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost
                                                :1521:orcl", "scott", "tiger");

        //Prepare query calling PL/SQL function
        String query = "{ ? = call first_fx (?, ?) }";
        //Create CallableStatement object
        CallableStatement cs = con.prepareCall (query);
        //Register return, OUT parameters with JDBC types.
        cs.registerOutParameter (1, Types.INTEGER); //return parameter
        cs.registerOutParameter (3, Types.INTEGER);
        //Set values to IN parameters
        cs.setInt (2, no);
        //Execute PL/SQL function
        cs.execute();
        //Gather results from return, OUT parameters
        System.out.println ("cube value:" + cs.getInt (1));
        System.out.println ("square value:" + cs.getInt (3));
```

```
//close jdbc objects  
cs.close();  
con.close();  
}  
}  
}
```

### PL/SQL function in oracle (dealing with sql queries)

```
create or replace function emp_fx (stno number, endno number, name out  
varchar2, bsal out number) return varchar;  
as  
desg varchar2(20);  
begin  
select ename into name from emp where empno >= stno and empno <= endno;  
select sal into bsal from emp where empno >= stno and empno <= endno;  
select job into desg from emp where empno >= stno and empno <= endno;  
return desg; select ename, sal, job into name,bsal,desg from emp where  
empno >= stno and empno <= endno  
end;  
/  
/
```

### // CSFxTest2.java

```
public class CSFxTest2  
{  
    public static void main (String args[]) throws Exception  
    {  
        //read input values from cmd line args  
        int stno = Integer.parseInt (args[0]);  
        int endno = new Integer (args[1]).intValue ();  
        // create jdbc con obj  
        Class.forName ("oracle.jdbc.driver.oracleDriver");  
        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:  
                                         :1521:orcl", "scott", "tiger");  
        //create callablestatement object  
        CallableStatement cs = con.prepareCall ("{ ?= call emp_fx (?, ?, ?, ?)}");  
        cs.registerOutParameter (1, Types.VARCHAR);  
        cs.registerOutParameter (4, Types.VARCHAR);  
        cs.registerOutParameter (5, Types.INTEGER);  
    }  
}
```

```

// set values in parameters
cs.setInt(2, stno);
cs.setInt(3, endno);

// execute PL/SQL function
cs.execute();

// gather results from out parameters, return parameters
System.out.println("emp name is :" + cs.getString(4));
System.out.println("emp salary is :" + cs.getInt(5));
System.out.println("emp Desg is :" + cs.getString(1));

// close JDBC objects
cs.close();
con.close();

} // main

```

// class

```

//> javac CSFxTest2.java
//> java Csfxtest2 7920 8000

```

### Example application

PL/SQL function having both select and non select SQL queries

create or replace function test\_fx(no in number, cnt out number) return sys\_refcursor  
as  
my cur sys\_refcursor;  
begin  
delete from emp where empno = no; → non select query  
cnt := SQL%ROWCOUNT; → holds a numeric value representing the no. of records  
open mycur for  
select empno, ename, sal, job from emp; → Here cursor holds bunch of  
selected records  
return mycur;  
end;
/

↑  
cursor as a return type

```

//CSTest3.java
import java.sql.*;
import java.util.*;
import oracle.jdbc.*;

public class CSTest3
{
    public static void main (String args[])
        throws Exception
    {
        // read input values from keyboard
        Scanner sc = new Scanner (System.in);
        System.out.println ("Enter employee no.");
        int no = sc.nextInt();

        // create jdbc connection object
        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection Con = DriverManager.getConnection ("jdbc:odbc:oradsn",
            //use type4 driver instead of type1
            "scott", "tiger");

        //create CallableStatement object
        CallableStatement cs = Con.prepareCall ("{ ? = call test_fix(?,?) }");

        //register OUT, return parameters with jdbc types
        cs.registerOutParameter (1, OracleTypes.CURSOR);
        cs.registerOutParameter (3, Types.INTEGER);

        //set values to IN parameter
        cs.setInt (2, no);

        //call PL/SQL function
        cs.execute();

        //gather results from OUT, return parameters
        ResultSet rs = (ResultSet) cs.getObject (1); //from return parameter
        int count = cs.getInt (3); //from OUT parameter

        //process the result
        if (count == 0)
            System.out.println ("Records not deleted");
        else
            System.out.println ("Count + " records deleted");
    }
}

```

```

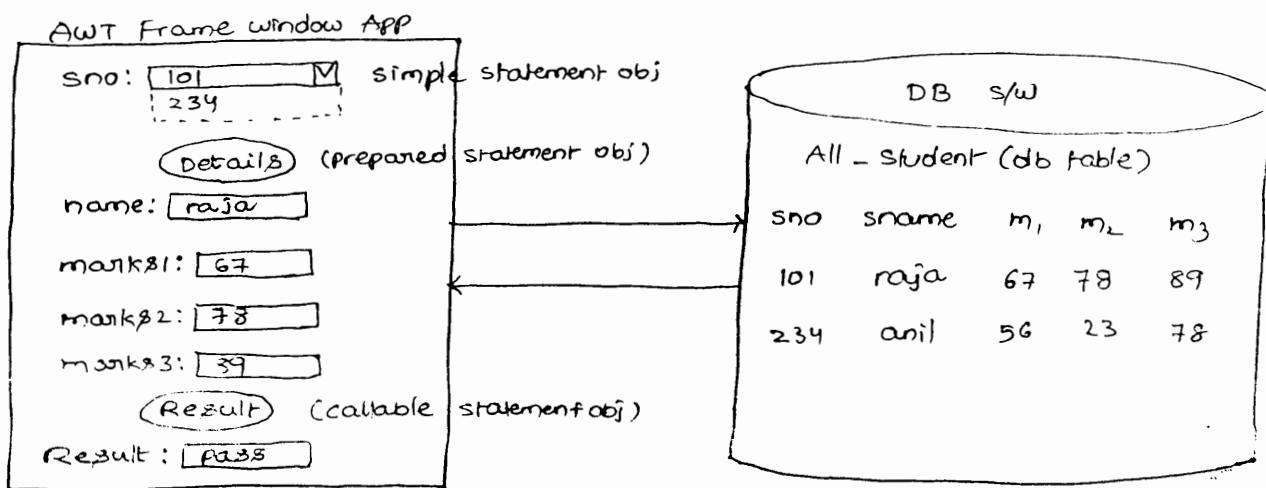
        while (ois.next())
    {
        System.out.println(ois.getInt(1) + " " + ois.getString(2) +
                           " " + ois.getInt(3) + " " + ois.getString(4));
    }
    //close jdbc objects
    cs.close();
    con.close();
}
//main
} //class

```

- \* All jdbc drivers give support to work with all the three types of jdbc statement objects, but type 1 driver can't use CallableStatement object to call cursor based PL/SQL functions and procedures.

#### Conclusion on three jdbc statement objects

- \* use simple statement object to execute SQL query only for one time in the entire application.
- \* use PreparedStatement object to execute same SQL query for multiple times with same or different values by making that query as pre-compiled query.
- \* To make persistence logic as centralized and reusable persistence logic take the support of PL/SQL functions and procedures of database software and use jdbc callablestatement object to call them from java applications.
- \* Using AWT, swing concepts we can develop GUI front end based applications to interact with database software.



\* By adding jdbc code in AWT, swing applications we can make them interacting with database software.

\* Example application to develop AWT based GUI front end application \*

Resources required in the above AWT App

#### Container

Frame window

#### Components

Label → 6

TextField → 5

choiceBox → 1

Button → 2

#### LayoutManager

FlowLayout

#### Events

ActionEvent

#### EventListener

ActionListener

#### Event handling method

actionPerformed( )

For the source code of above App

refer App 9 in booklet

page no: 22-24

\* In the above application all the three jdbc statement objects are used.

\* Student number to choice box should come only once in the entire execution of the application so its relevant query (select sno from student) should come be executed only once so simple statement object is recommended.

\* Based on the student number that is chosen from the choice box the remaining details of the student should come when details button is clicked. In this process we need to execute same sql select query (i.e. select \* from student where sno = ?) for multiple times with different values so PreparedStatement object is used.

\* To centralize the student result generalization logic PL/SQL procedure is taken in the db S/w so this PL/SQL procedure should be called when result button is clicked. For this CallableStatement object should be used.

- \* It is always recommended to create JDBC objects in one time execution block like constructor and utilize them in repeatedly executing blocks like event handling methods for better performance.
- \* If java application wants to talk with database software for 10 times instead of creating 10 no. of JDBC connection objects create one JDBC connection and use it for ten no. of times.
- \* Method called from constructor executes as one time execution block.
- \* self class methods and super class public/protected methods in sub class can be called without object.
- \* The logic that you want to execute automatically during application startup is called as load on startup logic. we need to place these logics either in static block (or) constructor.
- \* In the above diagram based application values in the choice box should be populated as load on startup values.
- \* Jdk supplies built-in tool called javaw to make the GUI Java Apps as windows applications/desktop applications.
- \* Basic jdk setup is enough to work with this tool.

#### Example:

Right click on desktop → new → short cut → javaw Allshmt&test →  
next → name for short cut collegeapp → finish → Right click on collegeapp icon → Properties → short cut tab → start in Application path  
short cut key Ctrl+Alt+Y  
→ change icon → apply → ok.

NOTE: The above javaw tool can not be used to make CUI applications as windows application.

#### How to execute CUI Java applications as windows based desktop application?

.. prepare batch file as shown below in any place of windows operating system.

myapp.bat (create on the desktop)

cd E:\NAPPS\APPS\advjava\jdbc

e:

javac selectTest.java

java selectTest

pause

what is the difference between AWT and swing?

### AWT

### swing

(1) Components look and feel is  
OS dependent

(1) OS independent

(2) Every Component maintains one  
peer component at OS level, so  
these components are heavy weight

(2) No peer component at OS level so  
these components are light weight

(3) gives minimum Components support  
(16+)

(3) gives more Components support  
(32+)

(4) Gives minimum Layout manager  
support (4)

(4) Gives more Layout manager support  
(6+)

Procedure to develop Allstmts application as swing application by using  
the NetBeans IDE 6.7

Step(1): create java project in netbeans IDE.

File menu → New Project → java → java application → next →  
Project Name: myproject → finish

Step(2): Add JFrame to the project source packages

Right click on source packages folder → new → JFrame Form  
class Name: AllstmtsTest → finish

Step(3): Design the frame window as shown in the previous diagram.  
by using drag and drop operations.

Step(4): Register ActionListener on button Components

Right click on details button → Events → action → actionPerformed

Perform the same operation on Result button.

Step(5): Declare the following instance variables in the source code of AllstmtsTest.java:

```
Connection con = null;  
Statement st = null;  
PreparedStatement ps = null;  
CallableStatement cs = null;  
ResultSet rs1 = null, rs2 = null;
```

Step(6) : write following user defined method in AllstmtsTest.java having jdbc code and call that method from constructor.

```
public void myinit()  
{  
    try  
    {  
        Class.forName("oracle.jdbc.driver.OracleDriver");  
        con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:  
                                         :1521:ord1", "scott", "tiger");  
        // load -on - startup logic  
        Statement st = con.createStatement();  
        rs1 = st.executeQuery("select sno from All-Student");  
        while (rs1.next())  
        {  
            JComboBox1.addItem((String) rs1.getString("sno"));  
        }  
        // create other jdbc statement objects  
        ps = con.prepareStatement("select * from All-Student where  
                               sno = ?");  
        cs = con.prepareCall("{ call FIND-PASS-FAIL (? ,? ,? ,? ) }");  
        cs.registerOutParameter(4, Types.VARCHAR);  
    }  
    catch (Exception e)  
    {  
        e.printStackTrace();  
    }  
} // myinit()
```

```

public Allstmt&Test() {
    Constructor
    {
        initComponents();
        myinit();
    }
}

```

step(7): write following code `jButtonActionPerformed (-)` method as details button related Event handling code.

```

private void jButton1ActionPerformed (java.awt.event.ActionEvent evt)
{
    //Code related details button

    try
    {
        int no = Integer.parseInt (comboBox1.getSelectedItem ());
        ps.setInt (1, no); (string) gives the selected item of  
comboBox
        //execute the query

        rs2 = ps.executeQuery();

        //Set the record of ResultSet object to text boxes
        if (rs2.next())
        {
            jTextField1.setText (rs2.getString (2));
            jTextField2.setText (rs2.getString (3));
            jTextField3.setText (rs2.getString (4));
            jTextField4.setText (rs2.getString (5));
        }
        rs2.close();
    }

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

```

step(8): write the following code in `jButton2ActionPerformed (-)` method for result button.

```

private void jButton2ActionPerformed (java.awt.event.ActionEvent evt)
{
    //for result button

    try
    {
        //Set IN parameters values
        cs.setInt (1, Integer.parseInt (jTextField2.getText()));
    }
}

```

```

cs.setInt(2, Integer.parseInt(jTextField3.getText()));
cs.setInt(3, Integer.parseInt(jTextField4.getText()));
//execute PL/SQL Procedure
cs.execute();
//gather result from OUT parameter and set result to textbox
jTextField5.setText(cs.getString(4));
}
catch (Exception e)
{
    e.printStackTrace();
}

```

Step(9): Add ojdbc14.jar file to the libraries of the project.

Right click on libraries folder → Add JAR → Browse and select  
ojdbc14.jar

Step(10): run the application

Right click on source code of AllStringTest.java → Run File

How to run a compiled java application from any folder of the project?

(a) Add the directory location of the application related .class file to classpath environment variable, to run that java application from any folder of the computer.

```

e:\apps\temp
    |
    +-- TESTAPP.java

```

TESTAPP.java

```

public class TESTAPP
{
    public static void main (String args[])
    {
        System.out.println("TESTAPP: main(-)");
    }
}

```

//>javac TESTAPP.java

• Add e:\apps\temp folder to CLASSPATH environment variable

• Go to Computer → Properties → Advanced tab → env variables →

variable name : CLASS PATH

value : e:\apps\temp; <other values>; .

→ ok → ok → ok

e:\> java TestAPP (success)

d:\> java TestAPP (success)

c:\xyz> java TestAPP (success)

\* Large scale companies prefer working with the commercial Oracle db s/w whereas small scale companies prefer working with the open source db s/w's like mysql, postgresql etc...

### MySQL

type : multi user db s/w

Vendor: Devx or sun microsystems (java soft)

open source or free db s/w

default portno : 3306

default username and password : root,root

(can be changed during installation/activation of s/w)

version: 4.x

Allows to create multiple logical databases

Default logical db names: mysql, test

To download software : www.devx.com

\* After installing MySQL db s/w we must activate it separately. For this

go to mysql\_home\bin and use winmysqladmin tool and look for traffic signal symbol in the system tray area.

Procedure to modify username and password of MySQL db s/w after installation

Right click on traffic signal symbol in system tray area → show me

→ my.ini setup tab → modify user and password → save modification

Procedure to create logical db, db table in mysql db s/w

Step(1): Launch the SQL prompt of MySQL database software

c:\mysql\bin → click on mysql.exe file

Step(2): create logical database

mysql> create database mydb1;

↓  
Name of the logical database

Step(3): Connect to this logical database <sup>to</sup> mydb1

mysql> connect mydb1;

Step(4): create db table in mydb1 logical db.

mysql> create table Employee (EID int(5), Primary key, FIRSTNAME  
varchar(20), LASTNAME varchar(20), EMAIL varchar(20));

Step(5): Insert records into this database table.

mysql> insert into employee values(345, 'raunesh', 'rao', 'rao@gmail.com')

\* If you are not aware of the SQL queries of certain database software then you can use some third party GUI based db tools to perform various operations on the db s/w.

Ex: mysql front for mysql db s/w

sqlyog for mysql db s/w

TOAD for mysql db s/w

TOAD for oracle db s/w

pl/sql developer for oracle db s/w.

NOTE: All the above tools must be installed ~~separately~~ separately.

Procedure to perform above given operation (like logical db creation and etc) by using sqlyog tool.

(1) Launch sqlyog tool

Launch → continue

MySQL Home address : localhost

User name : root

Password : root

Port : 3306

→ Connect

(2) Create new logical database.

right click on root@localhost → Create Database → mydb2 → Create

(3) Create database table in mydb2 logical database.

right click on mydb2 → Create Table → Enter values for table →  
Create Table → Table name: Employee

(4) Add records to the table.

right click on employee table → Insert/upgrade data → values

Procedure to develop Java application to interact with mysql db s/w by  
using type-1 jdbc driver:

Step(1): Install odbc driver for mysql separately in your computer (It  
will not come as default odbc driver).

Step(2): Create DSN for odbc driver for mysql.

Control panel → Performance and maintenance → Administrative Tools →  
Data Sources → User DSN → add → MySQL ODBC 5.1 Driver → Finish →

Data Source Name: mysqldsn

Server : localhost

Port : 3306

User : root

Password: root!

Database : mydb1

→ Test → OK

Step(3): Develop Java application as shown below by using type1 jdbc driver.

```
// SelectTest.java
import java.sql.*;
public class SelectTest
{
```

```

public static void main (String args[]) throws Exception
{
    //load jdbc driver class
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    //establish connection with db s/w
    Connection con = DriverManager.getConnection ("jdbc:odbc:mysqldsn",
                                                "root", "root1");
    //create Prepared Statement object
    PreparedStatement ps = con.prepareStatement ("select * from
                                                employee");
    //execute query
    ResultSet rs = ps.executeQuery();
    //process results
    while (rs.next())
    {
        System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3) +
                           " " + rs.getString(4));
    }
    //close jdbc objects
    rs.close();
    ps.close();
    con.close();
}
}

```

\* The Devx Company supplied type-4 mechanism based jdbc driver for MySQL is technically called as Connector/J jdbc driver.

### connector/J jdbc driver

target db s/w : mysql db s/w  
   version: 3.0.8  
   vendor: Devx  
 separate installation is required  
   driver mechanism : type 4  
 open source jdbc driver  
   jdbc driver class : org.gjt.mm.mysql.Driver (or)  
                          com.mysql.jdbc.Driver

JDBC URL: jdbc:mysql://<logical db name>

(or)

jdbc:mysql:/<hostname>:<portno>/<logical db name>

ZIP file that represents this JDBC driver: "mysql-Connector-Java-3.0.8-stable.ZIP".  
(download from devx.com).

JAR file that represents this JDBC driver: mysql-Connector-Java-3.0.8-stable-bin.jar file.  
(Collect from the above ZIP file extraction).

Procedure to work with Connector/J JDBC driver in Java application to interact with MySQL database software

Step(1): Gather the above said JAR file mysql-Connector-Java-3.0.8-stable-bin.jar file and add in the CLASSPATH.

Step(2): Gather Connector/J JDBC driver details as shown above.

Step(3): Develop Java application by using Connector/J JDBC driver as shown below.

```
//SelectTest.java
import java.sql.*;
public class SelectTest
{
    public static void main(String args[])
        throws Exception
    {
        //load JDBC driver class
        Class.forName("org.gjt.mm.mysql.Driver");
        //or
        //Class.forName("com.mysql.jdbc.Driver");
        //establish Connection with DB s/w
        Connection con=DriverManager.getConnection("jdbc:mysql://mydb",
                                                     "root", "root");
        //or
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:
3306/mydb", "root", "root");
```

3 3 =

\* To work with JDBC driver of any database software supplied by any vendor we must gather the following details.

- (i) JDBC driver class name that implements java.sql.Driver interface
- (ii) JDBC URL (or) DB URL to locate database software.
- (iii) Jar file that represents JDBC driver.

\* While developing small scale projects having the installation of MS Office take the support of MS Access as database software.

### MS-ACCESS

Type: single user db s/w

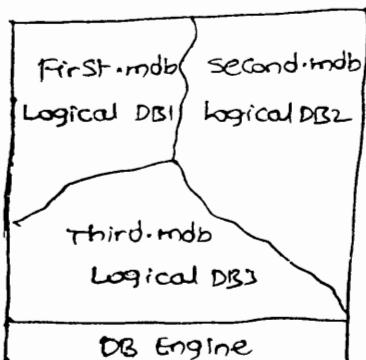
Version: 11.0 (MS-Office 2003)

Vendor: Microsoft

Commercial software

Allows to create logical databases

Each .mdb file acts as one logical database.



(Physical MS-ACCESS DB S/W)

### Procedure to create logical database and database table in MS-Access db s/w

Start → Programs → MS-Office → MS-ACCESS → Create new file → Blank database → Filename: mydb → Create → Create Table in design view

Field Name	Data type
Pid	Number
Pname	Text
Qty	Number

→ Ctrl + W → Yes → OK → NO → Launch Product Table → Enter data

as records → click on save button

\* Industry generally prefers working with type-1 jdbc driver to interact with MS-Access database software because the type-2 and type-4 jdbc drivers of MS-Access db s/w are commercial jdbc drivers.

Procedure to develop java application to interact with MS-Access db s/w

Step(1): create dsn for odbc driver for MS-Access.

control panel → performance and maintenance → administrative tools →

datasources → add → Microsoft Access Driver → Finish →

Data Source Name: accdsn

→ select → Browse and select the mydb.mdb file → ok

Step(2): Develop java application by using type-1 driver as shown below.

```
//AccessTest.java
```

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

```
public class AccessTest
```

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

```
{
```

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

```
    Connection con=DriverManager.getConnection("jdbc:odbc:accdsn");
```

"Be dsn created above"

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

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

```
while(rs.next())
```

```
{
```

```
    System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getInt(3));
```

```
}
```

```
rs.close();
```

```
st.close();
```

```
con.close();
```

```
} //main
```

```
} //class.
```

\* The single user of MS-Access is nothing but the windows login user.

### 3 overloaded forms of DriverManager.getConnection() method

- 1) Public static Connection getConnection (String url, String username, String password) throws Exception.
- 2) Public static Connection getConnection (String url);
- 3) Public static Connection getConnection (String url, Properties pro)

↓  
Map datastructure of  
java.util package

\* Use (1) and (3) form to interact with multiuser db s/w's (like oracle...).

\* Use (2) form to interact with single db softwares (like MS-Access).

### understanding java.util.Properties class

\* It is a map datastructure.

\* It is subclass to java.util.Hashtable.

\* It allows only string objects as keys and values of elements.

(Other map data structures allows any objects as keys and values in elements).

\* It allows the programmer to gather the element values from outside the application as text properties file content.

abc.txt (text Properties file)

name = ramesh
age = 30
address = hyd
It is demo file

The text file that maintain the entries in the form of key:value pairs is called as text properties file.

Example application to create java.util.Properties class object by utilizing the data of text properties file.

```
e:\app8
  |-adv.java
    |-idbc
      |-myfile.properties
      |-Prop8Test.java
```

### myfile.properties

```
# Demo file
name = raja
age = 30
address = hyd
```

### Propstest.java

```
// Propstest.java
import java.io.*;
import java.util.*;
public class Propstest
{
    public static void main(String args[]) throws Exception
    {
        //Locate properties file
        FileInputStream fis = new FileInputStream ("myfile.properties");
        //create Properties class object and load myfile.properties data
        //into that object
        Properties p = new Properties();
        p.load(fis);

        System.out.println("Data of Properties class obj is :" + p.toString());
        System.out.println("Data value of name key is :" + p.getProperty("name"));
    }
}
```

p(Properties class object)

keys	values	
name	raja	(element-0)
age	30	(element-1)
address	hyd	(element-2)

## Example on (3) form of DriverManager.getConnection (-) method.

e:\apps  
    | adv.java  
    |     | jdbc  
    |     |     | DbDetails.properties  
    |     |     | SelectTest.java

### DbDetails.properties

# DB details  
{ user = scott  
  | password = tiger  
  |  
  | fixed keys  
  | as given in documentation

### SelectTest.java

```
//SelectTest.java
import java.io.*;
import java.sql.*;
import java.util.*;

public class SelectTest
{
    public static void main (String args[]) throws Exception
    {
        //locate Properties file
        FileInputStream fis = new FileInputStream ("dbdetails.properties");
        //Store dbdetails.properties file data into java.util.Properties class
        //object
        Properties p = new Properties();
        p.load(fis);
        //register jdbc driver with DriverManager service
        Class.forName ("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@local
                                                host:1521:orcl", p);
        =====
        ;
        //javac SelectTest.java
        //java SelectTest
```

## Postgresql

Type: multi-user db s/w

Version: 8.3

Vendor: university of California & Mr. Scott.

open source db s/w

default portno: 5432

To download s/w: download as zip file from www.postgresql.org website (postgresql-8.3.13-1.zip)

\* To install postgresql software extract postgresql-8.3.13-1.zip file to a folder and use the setup file that comes after extraction.

Procedure to create logical database and database table in postgresql

In database software (in the default logical database postgres)

Step(1): Start postgresql database software.

Start → Programs → postgresql 8.3 → Start Service

Step(2): Open sql prompt of postgresql database software.

Start → Programs → postgresql → psql to postgres → password for user postgres: sathyas

↳ chosen during installation

NOTE: Here we are connected to default logical database called postgres

Step(3): Create database table

```
create table product (Pid int, Pname varchar(20), Price real);
```

Step(4): Insert records into database table

```
insert into product values (101, 'Table', 678.45);
```

```
insert into product values (102, 'Pen', 348.45);
```

```
commit;
```

```
select * from product;
```

Pid	Pname	Price
101	Table	678.45
102	Pen	348.45

allows to create logical dbs the default logical db is postgres

- \* The type4 mechanism based third party vendors like jdbc drivers for postgresql db s/w is technically called as postgresql thin driver.
- \* The type4 mechanism jdbc drivers are called as thin drivers whereas the type1 and type2 mechanism based drivers are called as thick drivers.

### Postgres thin driver (type4)

target db s/w: Postgres db s/w

mechanism : type4

Vendor : Open Community (University of California)

open source jdbc driver

jdbc driver class: org.postgresql.Driver

URL : jdbc:postgresql:<logical db name>

jar file that represents jdbc driver: postgresql-8.4-701.jdbc3.jar

(for jdbc3 spec based jdbc drivers)

postgresql-8.4-701.jdbc4.jar

(for jdbc4 spec based jdbc driver)

### Procedure to develop java application interacting with postgresql db s/w

#### by using the above thin driver

Step(1): Arrange the postgresql database software related thin driver based jar file in a folder.

E:\postgresql\jarfiles

  |→ postgresql-8.4-701.jdbc4.jar

    (for jdk 1.6 environment)

  |→ postgresql-8.4-701.jdbc3.jar

    (for jdk 1.5 environment)

Step(2): Add one of the above given two jar files to classpath.

Step(3): Develop java application as shown below by using the postgresql thin driver.

```
→ PostgreSQLApp.java
import java.sql.*;
```

```

public class PostgreSQLApp
{
    public static void main (String args[])
        throws Exception
    {
        //create jdbc con
        Class.forName("org.postgresql.Driver");
        //Connection Con = DriverManager.getConnection ("jdbc:postgresql://
        // or
        Connection Con = DriverManager.getConnection ("jdbc:postgresql://
            localhost:5432/postgres",
            "postgres", "satya");

        //create jdbc statement object
        Statement st = Con.createStatement();
        //send and execute SQL query
        ResultSet rs = st.executeQuery ("Select * from product");
        //process the result
        while (rs.next())
        {
            System.out.println (rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
        }
        rs.close();
        st.close();
        Con.close();
    }
}

```

Step (4): compile and execute the java application.

\* Latest specification of jdbc is : jdbc 4.0

\* Each that is developed based on jdbc specification contains version.

    type1 version: 2.x

    Oracle thin driver version: 9.2.x

    Postgres thin driver version: 8.4.x

\* JEE module is not a installable software, it is a big s/w specification having lots of sub specifications like servlet specification, JSP specification, EJB specification and etc.

\* Latest JEE specification is JEE5

In that

  servlet 2.5 spec

  JSP 2.1 spec

  EJB 3.0 spec

  JMS 1.2 spec are there as sub specifications.

### Meta-data

\* Data about data is called as meta data that means gathering much information about given information is nothing but meta-data.

\* If data is one website then search engine website looks like meta data.

\* In jdbc meta data programming is useful to know limitations, capabilities and facilities of underlaying database software and its resources.

\* JDBC supports 3 meta data programming concepts.

- 1) DatabaseMetaData
- 2) ResultSetMetaData
- 3) ParameterMetaData

### Database MetaData :

\* DatabaseMetaData object means it is the object of a class that implements "java.sql.DatabaseMetaData" interface.

\* By invoking various methods on DatabaseMetaData object we can gather the following details.

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

Method declared in java.sql.Connection interface

→ Database name and version

→ support for PL/SQL procedures and functions

→ Max characters in table name and column name etc...

### ResultSetMetaData

\* It gives various details about the database table that is represented by JDBC Resultset object like column names, column datatypes, column count and

etc...

```
ResultSet rs = st.executeQuery("Select * from student");
ResultSetMetaData rsmd = rs.getMetaData(); // method declared in
                                         java.sql.ResultSet
```

\* ResultSetMetaData object means it is the object of a class that implements "java.sql.ResultSetMetaData" interface.

#### ParameterMetaData:

\* Gives details about parameters available in sql queries of jdbc PreparedStatement object and callable statement object.

PreparedStatement ps = Con.prepareStatement ("insert into student values

(?, ?, ?)");

↓  
parameters of the query

ParameterMetaData pmd = ps.getParameterMetaData();

\* ParameterMetaData object means it is the object of a class that implements "java.sql.ParameterMetaData" interface.

\* Using pmd object we can gather the details of parameter like parameter name, type and other details.

\* Metadata programming is no way related with performing CURD operations on database tables. It is given to gather advanced details about database software and other resources.

For example application on DataBaseMetadata refer application 16 of the booklet (page no-29)

\* If any method calls of metadata programming returns 0 or null then we need to understand the underlying jdbc driver is not capable of gathering that information.

\* The metadata programming related output will change based on the database software and the jdbc driver we use.

\* DatabaseMetadata is useful to develop certain database tables like SQL DB tools like sql yug, mysql front, toad etc...

```

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

        Connection con= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:ordl", "scott", "tiger");

        Statement st = con.createStatement();

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

        while(rs.next())
    }
}

```

\* The above code gives db table (student records) but in order to display them with column names and values take the support of "ResultSetMetadata" object.

Example code to print ResultSet object data along with column names and column datatypes

```

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

        Connection con= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:ordl", "scott", "tiger");

        Statement st = con.createStatement();

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

        ResultSetMetaData rsmd = rs.getMetaData();

        int colcnt = rsmd.getColumnCount();

        // to print column names

        for(int i=1; i<colcnt; i++)
    {
        System.out.println(rsmd.getColumnLabel(i) + "\t\t\t");
    }

    System.out.println();
    // to print datatypes

    for (int i=1; i<=colcnt; i++)
    {
        System.out.println(rsmd.getColumnTypeName(i) + "\t\t\t");
    }

    System.out.println();

```

```

    while(rs.next())
    {
        sop(rs.getInt(1) + "\t\t\t" + rs.getString(2) + "\t\t\t" +
            rs.getString(3));
    }
    rs.close();
    st.close();
    con.close();
}

* ResultSet Metadata programming is quite useful towards reports generation
on database table data.

```

How can you retrieve database table records by passing table name to application dynamically (without knowing columns)?

use ResultSetMetadata object as shown below

```

import java.sql.*;
public class xyz
{
    public static void main (String args[])
        throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:
                                                1521:orcl", "scott", "tiger");
        Statement st = con.createStatement ();
        ResultSet rs = st.executeQuery ("Select * from " + args[0]);
        ResultSetMetadata rsmd = rs.getMetaData ();
        int colcnt = rsmd.getColumnCount ();
        // to print column names
        for (int i=1; i<=colcnt; i++)
        {
            sop(rsmd.getColumnName(i) + "\t\t");
        }
        sop();
    }
}

```

```

//to print col datatypes
for (int i=1 ; i<=colcnt ; i++)
{
    sop(rsmd.getColumnName(i) + "\t\t");
}
sop();
//to display records
while(rs.next())
{
    for (int i=1 ; i<=colcnt ; i++)
    {
        sop(rs.getString(i) + "\t\t");
    }
    sop();
}
rs.close();
st.close();
con.close();
}
}

```

<u>O/P</u>	EId	Ename	Add	
	Number	Varchar	Varchar	getColumnName
	101	Raja	hyd	getColumnType
	235	Ramesh	hyd	getName

getString

NOTE: while executing the above code pass database table name as Command Line argument.

what is the difference between Resultset object and Result-set Metadata object ?

\* Resultset object points to table data whereas Resultset metadata object points to database table structure like column names, column data types, and etc...

\* Most of the jdbc drivers are not giving support for parameter metaData programming.

+ The maximum vendors of jdbc drivers are not providing implementation for "java.sql.ParameterData" interface.

## DB S/W

### 1. Conventional DB S/W

oracle  
sybase  
MS-Access  
MySQL  
Postgresql

### 2. Non-conventional DB S/W

MS-Excel  
CSV-files  
Tab-files

\* Non-conventional DB S/W will be used as helping/supporting DB S/W

when Conventional DB S/Ws are used as main DB S/Ws.

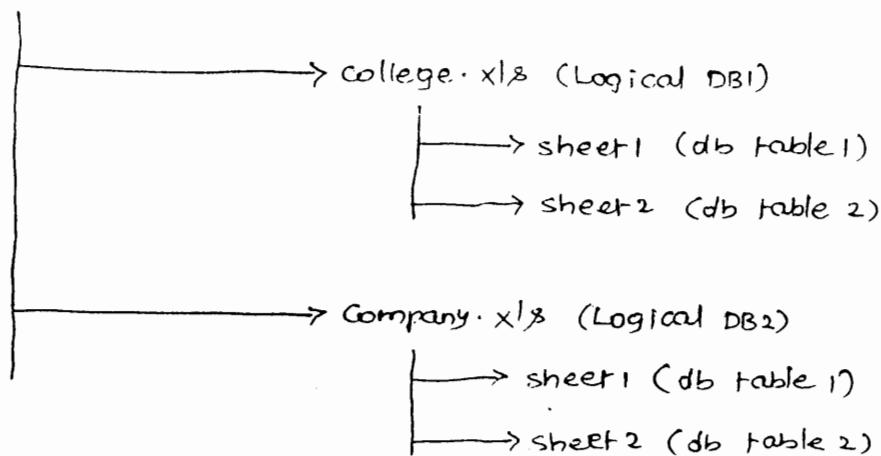
\* In ICICI banking project oracle is taken as Main DB S/W and MS-Excel is taken as helper DB S/W to get the printouts of accounts statements after every 3 months for this they move last 3 months data from oracle to MS-Excel.

\* MS-Excel gives good performance towards print operations when compare to oracle.

\* Since ODBC drivers are already available to communicate with all non-conventional DB S/Ws. So use JDBC driver type-1 for java application to non-conventional DB S/W communication.

### MS-Excel :

#### MS-Excel S/W (Physical DB S/W)



sheet1 of college.xls

sno	sname	add
101	Ravi	hyd
345	Anil	hyd
123	Rajesh	Vi3ag

← (dbtable column names)

db table col values  
db tables records

\* Here College.xls and Company.xls are workbooks

Procedure to interact with MS-Excel from java application

Step(1): Create work book in MS-Excel s/w (College.xls as shown above)

E:\APP\\$ \temp\College.xls file

Step(2): Create DSN for the Microsoft Excel driver

Control panel → performance and maintenance → Administrative tools →

Data Sources → User DSN tab → Add → Microsoft Excel (.xls) →

Finish →

Data Source Name : xls dsn (anyname)

→ select work book → browse and select → E:\APP\\$ \temp\College.xls → ok

Step(3): Develop java application by using type1 jdbc driver as shown below

// Excel Test.java

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

```
public class ExcelTest
```

```
{
```

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

```
    {
```

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

```
        Connection con = DriverManager.getConnection("jdbc:odbc:xlssdsn");
```

(Excel is single user db so no  
need to username and password)

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

```
        ResultSet rs = st.executeQuery("select * from [sheet1$]");
```

The MS-Excel sheet name as db table name

```

        while(gis.next())
    {
        sop(gis.getInt(1)+" "+gis.getString(2)+" "+gis.getString(3));
    }
    gis.close();
    st.close();
    con.close();
}

}//main
}//class

```

CSV - file's (comma separator values) :

E:\apps

→ temp (Logical DB)

→ abc.csv (db Table 1)  
→ myfile.tab (db Table 2)

abc.csv

sno, sname, add → db Table Column names		
101	Raja	hyd
345	Rajesh	hyd

} db Table records

myfile.tab

sno sname add → Column names		
101	Raja	hyd
345	Ramesh	hyd

} db Table records

\* The total file system of OS (Windows) acts as physical DB s/w, in that each directory acts as one logical DB, in that directory each file that contains formatted data acts as one DB table.

Procedure to communicate with text file (csv file) from java application

Step 1: Prepare the csv file ready as shown above

E:\NAPPS\apps\temp\abc.csv

Step 2: Create DSN for Microsoft Text Driver

Control panel → Performance and maintenance → Administrative tools →

Data Sources → User DSN tab → Add → Microsoft Text driver → Finish

Data Source Name : txtdsn (any name)

→ Select directory → E:\NAPPS\apps\temp\ → OK

Step(3): Develop Java application by using type-1 driver as shown below.

```
// TextTest.java
import java.sql.*;
class TextTest
{
    public static void main (String args[]) throws Exception
    {
        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection Con= DriverManager.getConnection ("jdbc:odbc:txtdsn");
        Statement st = Con.createStatement ();
        ResultSet rs = st.executeQuery ("select * from abc.csv");
        while (rs.next ())
        {
            System.out.println (rs.getInt (1) + " " + rs.getString (2) + " " + rs.getString (3));
        }
        rs.close ();
        st.close ();
        Con.close ();
    } //main
} //class
```

↓  
filename as db table name

\* In one JDBC application there can be multiple connection objects, multiple statement objects and multiple Resultset objects.

\* One JDBC application can interact with multiple DB softwares at a time.

Write a program to transfer the records of MS Excel to oracle DB S/w

Refer Application 13 of Page no 27, 28

### Batch processing:

\* The process of combining related queries into single unit, sending that unit to DB S/w, executing queries of that unit in DB S/w and gathering results of those queries as a single unit is called as "batch processing".

- \* Batch processing reduces network round trips between java application and database software especially when java application wants to send and execute multiple SQL queries in database software.
- \* Instead of sending multiple queries to database software individually one by one it is recommended to take the support of batch process.
- \* The query is placed in batch will not be executed by enabling do everything or nothing principle (that means if one query execution fails the remaining queries executions should also be failed/rollback).

### Procedure to perform JDBC batch processing

- (1) create jdbc statement object

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

- (2) Add non select queries to batch.

Select queries can't be added to the batch of batch processing

```
{ st.addBatch("insert into student values(111, 'xxx', 'yyy')");
  st.addBatch("update student set saddr = "hyd" where sno=101");
  st.addBatch("update student set add = "vizag" where sname = "Raju\"");
  st.addBatch("delete from student where sno = 111");}
```

→ keep all these SQL queries in a single batch.

NOTE: Any no. of non-select queries pointing to same database table or different database tables can be added to the above batch for batch processing.

- (3) Send and execute the above batch of queries in database software.

```
int res[] = st.executeBatch();
```

res[]

1	0
1	1
0	2
3	3

This `int res[]` contains the batch related queries execution results.

- \* select query execution gives JDBC ResultSet object, executeBatch() returns int[] and ResultSet object can't be stored in this int[] so we can't add select SQL query to the batch of batch processing.
- \* for example application on batch processing refer page no 31 and application 21.
- \* The process of combining related operations into a single unit and executing them by applying do everything or nothing principle is comes under transaction management.

Ex: Transfer money task is the combination of 2 operations.

- withdraw amount from source account
- Deposite amount into destination account.

\* we need to execute these two operations by applying do everything or nothing principle performing transaction management.

+ To perform transaction management in JDBC applications we use batch processing along with some additional logics.

#### Transaction Management in JDBC:

\* The process of combining set of related operations into single unit and executing those operations by applying "do everything or nothing" principle is called as transaction management.

#### Example application (performing transfer money operation)

SQL> select \* from Account tab;

Ac no	Holdeurname	Balance
101	Raja	10,000
102	Ravi	30,000

keep DB table ready as shown in the table.

develop the application to transfer the funds between two accounts.

// TxMgmt.java

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

```

//read input values from keyboard
Scanner sc = new Scanner (System.in);
scop("Enter source account no:");
int s1cno = sc.nextInt();
scop("Enter destination account no:");
int desno = sc.nextInt();
scop("Enter the amount to transfer");
int amt = sc.nextInt();
//create jdbc connection object
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:
1521:ord", "scott", "tiger");
//Disable autoCommit mode (beginning of transaction (Tx))
con.setAutoCommit (false);
//perform transfer money operation (through batch processing)
Statement st = con.createStatement ();
//for withdraw operation on source account
st.addBatch ("update Account_tab set balance = balance - " + amt + " where
acno = " + s1cno );
//for deposit operation on destination account
st.addBatch ("update Account_tab set balance = balance + " + amt + " where
acno = " + desno );
//execute the batch
int gres [] = st.executeBatch ();
//write logic to Commit or rollback the transaction
boolean flag = false;
for (int i = 0; i < gres.length; i++)
{
    if (gres [i] == 0)
    {
        flag = true;
        break;
    }
}
if (flag == true)
{
    con.rollback ();
    scop (" Transaction rolledback");
}
else
    con.commit ();

```

```

        sop ("Transaction committed");

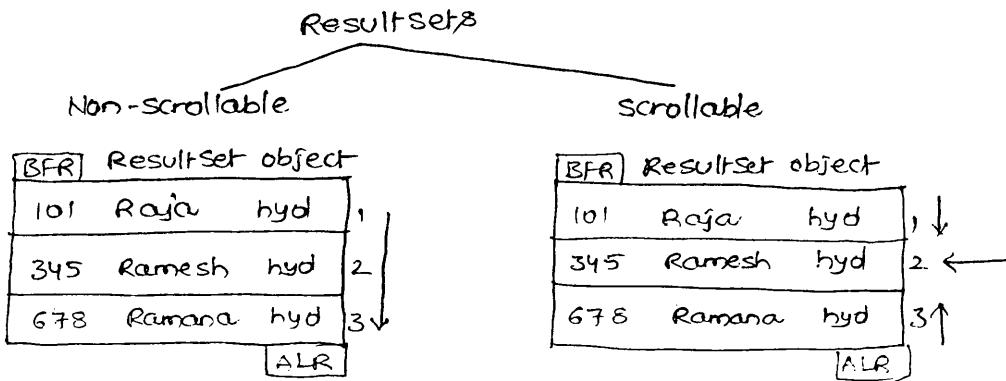
    }

    st.close();
    con.close();
} //main
} //class

```

\* Batch processing is not possible with preparedstatement object and callablestatement object.

### working with different types of Resultsets :



- \* The Resultset object that allows to access the records sequentially, unidirectionally (top to bottom) is called as non-scrollable object.
- \* By default every jdbc resultset object is non-scrollable resultset object.
- \* The Resultset object that allows the programmer to access records randomly or non-sequentially or directly or bidirectionally is called as scrollable Resultset object.
- \* Using scrollable Resultset object we can access the records of Resultset object quickly.

### To create non-scrollable Resultset object

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

```
ResultSet rs = st.executeQuery("Select * from student");
          ↓
Non-scrollable Resultset object
```

### To create scrollable Resultset object

```
Statement st = con.createStatement(type, mode);
```

```
ResultSet rs = st.executeQuery("Select * from student");
          ↓
Scrollable Resultset object.
```

## Possible values for type (int):

ResultSet. TYPE\_SCROLL\_SENSITIVE ; (1005)

ResultSet. TYPE\_SCROLL\_INSENSITIVE ; (1004)

## Possible values for Mode (int):

ResultSet. CONCUR\_UPDATABLE ; (1008)

ResultSet. CONCUR\_READ\_ONLY ; (1007)

NOTE: The possible values of type, mode parameters are int constants internally in java.sql.ResultSet interface.

\* Public static final member variables of java class/ java interface are called as constants.

\* The jdbc statement object that is created without type and mode values can create only non-scrollable Result set object.

The jdbc statement object that is created having type and mode values can create scrollable Result set object.

The methods invokable on non-scrollable Result set object

next();

getXXX();

getRow();

close();

The methods invokable on scrollable Result set object

next();

getXXX();

`getRow();`

`close();`

`previous();`

`beforeFirst();` → keeps cursor in BFR position

`afterLast();` → keeps cursor in ALR position  
↓  
Record pointer.

`first();` → keeps cursor in first record

`last();` → keeps cursor in last record

`isFirst();` → checks whether cursor is there in first record or not.

`isLast();` → checks whether cursor is there in last record or not.

`absolute (-/+n);`

`relative (-/+n);`

what is the difference between `absolute (-)` and `relative (-)` methods?

\* `absolute (-)` method moves the record pointer in resultset object in forward and reverse direction. positive no. indicates forward direction with respect to first record. Negative no. indicates with respect to last record.

Ex: `absolute(3)` → moves the cursor to (from first) 3rd record  
`absolute(-2)` → moves the cursor to 2nd record from last.

\* `relative (-)` method moves the cursor (record pointer) with respect to current position of cursor. positive no. indicates forward direction, negative no. indicates reverse direction.

What is the difference between absolute path and relative path?

e:\apps

    |  
    |→ Demo

    |  
    |→ abc.txt

The absolute path of abc.txt file is

e:\apps\Demo\abc.txt

The relative path of abc.txt file from e:\apps folder

..\\Demo\\abc.txt

The relative path of abc.txt file from e:\apps\xyz folder

..\\..\\Demo\\abc.txt

The relative path of abc.txt file from e:\apps\demo\abc folder

..\\abc.txt

.. → indicates current directory

.. → indicates parent directory

\* In relative path the file/directory location will be given with respect to the current working folder location.

\* In absolute path the directory or file location will be given completely with respect to right from root directory.

---

\* To display the records of JDBC resultset object bidirectionally (top to bottom and bottom to top) we need to work with scrollable resultset object as shown below.

```
// ScrollTest.java
```

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

```
public class ScrollTest
```

```
{
```

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

```
    {
```

```
        // Create JDBC Connection object
```

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

```
        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:orcl", "scott", "tiger");
```

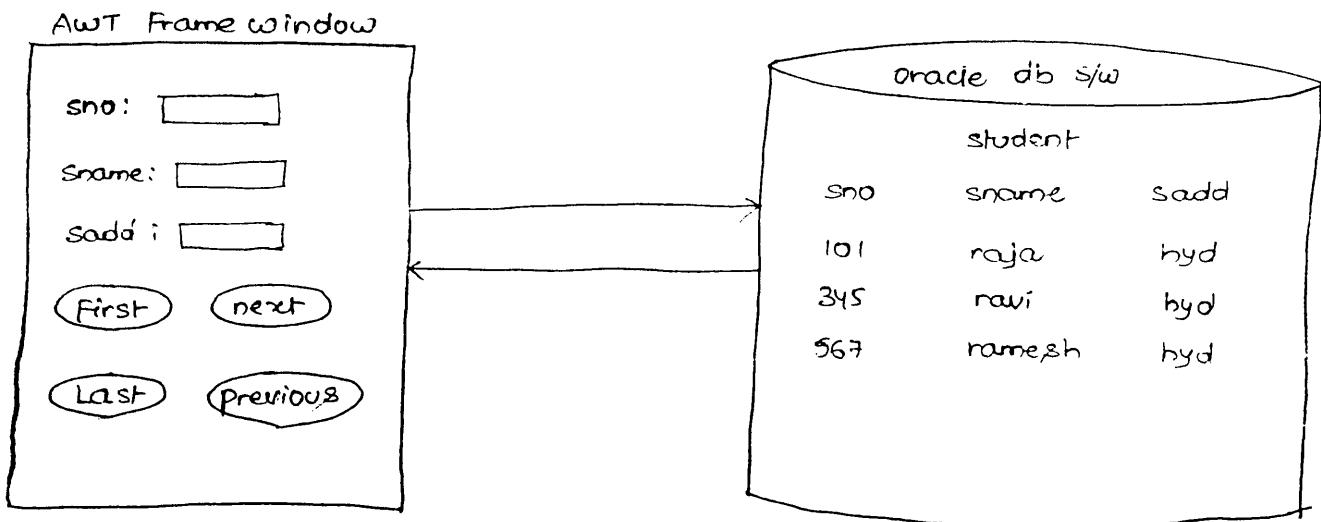
```
// create jdbc statement object with type and mode values  
Statement st = Con.createStatement (ResultSet.TYPE_SCROLL_SENSITIVE,  
                                  ResultSet.CONCUR_UPDATABLE);  
                                         L→Type (1005)  
                                         L→mode (1008)  
// get scrollable Resultset object  
  
ResultSet rs = st.executeQuery ("select * from student");  
// print records in bottom to top order  
  
rs.afterLast(); // keeps cursor in ALR position  
  
while (rs.previous ())  
{  
    System.out.println (rs.getInt (1) + " " + rs.getString (2) + " " + rs.getString (3));  
}  
  
// print records in top-bottom order  
  
rs.beforeFirst(); // keeps cursor in BFR position  
  
System.out.println ("records... (top to bottom)");  
  
while (rs.next ())  
{  
    System.out.println (rs.getInt (1) + " " + rs.getString (2) + " " + rs.getString (3));  
}  
  
// 3rd record is  
  
rs.absolute (3);  
  
System.out.println (rs.getRow () + " record is: " + rs.getInt (1) + "  
                         + rs.getString (2) + " " + rs.getString (3));  
  
// last but 2 record  
  
rs.absolute (-2);  
  
System.out.println (rs.getRow () + " record is: " + rs.getInt (1) + "  
                         + rs.getString (2) + " " + rs.getString (3));  
  
// 4th record from the bottom  
  
rs.relative (-2);  
  
System.out.println (rs.getRow () + " record is: " + rs.getInt (1) + "  
                         + rs.getString (2) + " " + rs.getString (3));  
  
// last but one record  
  
rs.relative (3);  
  
System.out.println (rs.getRow () + " record is: " + rs.getInt (1) + " " +
```

```

    rs.getString(2) + " " + rs.getString(3));
    //The first record is
    rs.first();
    System.out.println(rs.getRow() + "record is!" + rs.getInt(1) +
                       + rs.getString(2) + " " + rs.getString(3));
    //the last record is
    rs.last();
    System.out.println(rs.getRow() + "record is!" + rs.getInt(1) +
                       + rs.getString(2) + " " + rs.getString(3));
    //close jdbc objects
    rs.close();
    st.close();
    con.close();
}
//main
}
//class

```

- \* Non scrollable Resultset objects are forward only resultset objects by default. That means they allow the programmer to move the cursor in a resultset object only in forward direction. (top to bottom)
- \* All jdbc drivers support scrollable resultsets.



Container → frame window

Components → label - 3, Button - 4, TextField - 3

LayoutManager → FlowLayout

Event → Action Event

EventListener → ActionListener

Event handling method → actionPerformed (-)

\* For the above diagram based application refer application 10 of page nos 24-26.

Procedure to develop scrollFrame application (given in previous class) as an awt application by using netbeans IDE.

Step(1): Create java project in netbeans IDE

File menu → new → java → Java Application → next → myproject1 → Finish.

Step(2): Add awt frame to the project

Right click on project → new → other → AWT GUI form → Frame form → next →

classname: scrollFrame

→ Finish

Step(3): Design frame window as shown in the diagram with awt components.

NOTE: Before performing above step operation change the Layout of frame window from the default BorderLayout to NullLayout  
[inspector window → right click on BorderLayout → Free design/NullLayout]

Step(4): Register ActionListener on these button Components.

Right click on button (any button) → Events → Action → actionPerformed  
(do this on all 4 buttons)

Step(5): Declare the following instance variables in the source code of scrollFrame class.

```
Connection Con=null;  
ResultSet rs=null;  
Statement st=null;
```

Step(6): Develop user defined method in ScrollFrame class having JDBC code to create scrollable ResultSet object.

```
public void makeConnection()  
{  
    try  
    {  
        Class.forName("oracle.jdbc.driver.oracleDriver");  
        Connection Con = DriverManager.getConnection("jdbc:oracle:thin:  
                                         @localhost:1521:orcl", "scott", "tiger");  
        st = Con.createStatement (ResultSet.TYPE_SCROLL_INSENSITIVE,  
                               ResultSet.CONCUR_UPDATABLE);  
        rs = st.executeQuery ("Select * from student");  
    }  
    catch (Exception e)  
    {  
        e.printStackTrace();  
    }  
}
```

// makeConnection

Step(7): call the above makeConnection method from the constructor.

```
public scrollFrame()  
{  
    initComponents();  
    makeConnection();  
}
```

Step(8): Add following logic in button1<Actionperformed>method (for first button)

```
private void button1ActionPerformed (-)  
{  
    try  
    {  
        rs.first();  
        textField1.setText (rs.getString(1));  
        textField2.setText (rs.getString(2));  
        textField3.setText (rs.getString(3));  
    }  
}
```

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

Step(9): write following code in button2ActionPerformed (-) method  
(for next button)

```
private void button2ActionPerformed (-)
{
    try
    {
        if (!ois.isLast())
        {
            ois.next();
            textField1.setText(ois.getString(1));
            textField2.setText(ois.getString(2));
            textField3.setText(ois.getString(3));
        } // if
    } // try
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

Step(10): write following code in button3ActionPerformed (-) method  
(for previous button)

```
private void button3ActionPerformed (-)
{
    try
    {
        if (!ois.isFirst())
        {
            ois.previous();
            textField1.setText(ois.getString(1));
            textField2.setText(ois.getString(2));
            textField3.setText(ois.getString(3));
        } // if
    } // try
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

Step(11): write the following code in button1ActionPerformed (-) method  
(for Last button)

```
private void button1ActionPerformed (-)
{
    try
    {
        rs.last();
        textField1.setText(rs.getString(1));
        textField2.setText(rs.getString(2));
        textField3.setText(rs.getString(3));
    } //try
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

Step(12): Add ojdbc14.jar file to the libraries of the project.

Right click on libraries folder → Add jar → browse and select  
ojdbc14.jar file

Step(13): Run the application.

Right click in the source code of scrollFrame.java → Run file.

\* A scrollable ResultSet object can also be developed having some additional behaviors like SensitiveResultSet, InSensitiveResultSet, UpdatableResultSet, ReadonlyResultSet.

What is the difference between SensitiveResultSet object and InSensitiveResultSet object?

\* When ResultSet object is representing db table data if modifications done in db table are immediately reflected to ResultSet object then it is called as SensitiveResultSet object. Otherwise it is called as InSensitiveResultSet object.

To create SensitiveResultSet object

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

ResultSet rs = st.executeQuery("select \* from student");

### To create InSensitive Resultset object

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

ResultSet rs = st.executeQuery("select \* from student");

\* TYPE! JDBC driver supports Sensitive Resultsets but not InSensitive Resultsets.

\* Oracle thin driver, OCI driver, Connector/J driver of MySQL, PostgreSQL thin driver supports INSENSITIVE Resultsets but not SensitiveResultsets.

\* For example application on sensitive and InSensitive Resultsets refer application 15 of page no.s 28 and 29.

\* Don't use Sensitive and InSensitive Resultset object features in your JDBC based projects because no driver is supporting these sensitive and InSensitive behaviors properly.

### Creating Scrollable Resultset objects by using Prepared Statement object

```
PreparedStatement ps = con.prepareStatement("select * from student",
```

```
                                ResultSet.TYPE_SCROLL_SENSITIVE,  
                                ResultSet.CONCUR_UPDATABLE);
```

ResultSet rs = ps.executeQuery();

\* we can use simple Statement object or Prepared Statement object to work with scrollable Resultset object.

What is the difference between readonly resultset object and updatable resultset object?

\* If the modifications done in resultset object are reflecting immediately to database table that result set object is called updatable resultset object. Through updatable result set object we can perform all the four CURS operations on the database table.

The Resultset object that allows only read operations on database table is called as readonly resultset object.

what is the difference between sensitive resultset object and updatable resultset object?

\* In Sensitive resultset object the modifications done in database table will reflect to resultset object whereas in updatable resultset object the modifications done in resultset object will reflect to database table.

### TO prepare scrollable Resultset object as Readonly Resultset object

```
Statement st = con.createStatement (ResultSet.TYPE_SCROLL_SENSITIVE,  
                                  ResultSet.CONCUR_READ_ONLY);
```

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

### TO prepare scrollable Resultset object as updatable Resultset object

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

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

\* All non-scrollable resultset objects are Readonly Resultset objects by default.

\* we can perform both select and non-select operations on database table by using updatable resultset object as shown below

### TO perform select operation

```
while (rs.next ())  
{  
    System.out.println (rs.getInt (1) + " " + rs.getString (2) + " " + rs.getString (3));  
}
```

### TO insert new record

```
rs.moveToInsertRow (); → creates an empty record
```

```
rs.updateInt (1, 567);  
rs.updateString (2, "rajesh"); } sets values to the columns of the empty  
rs.updateString (3, "hyd"); record.
```

```
rs.insertRow (); → inserts records in database table
```

### TO update the existing record

```
rs.absolute (4); → moves to 4th record in Resultset object
```

```
rs.updateString (2, "rakesh"); → modifying 2nd column value of 4th record.
```

`rs.updateRow();` → updates the record in database table.

### To delete the existing record

```
rs.absolute(3);
```

```
rs.deleteRow();
```

\* Using updatable Resultset object we can perform insert, update and delete operations on database table without using the respective sql queries.

### \* Limitations with updatable Resultset objects \*

(1) Very few jdbc drivers support Updatable Resultsets.

(2) We can not perform criteria based non-select operations on database table.

(3) Bulk non-select operations are not possible.

NOTE: Only Type-1 jdbc driver is giving support for updatable result set objects. Oracle thin driver, OCI driver, MySQL Connector/J driver, PostgreSQL thin driver are not supporting updatable Resultsets.

\* All jdbc drivers are giving support for Readonly Resultset objects.

### Summary table on jdbc Resultset objects

#### Type-1 driver

Support for Non-scrollable Resultset : yes

Support for scrollable Resultset : yes

Support for Readonly Resultset : yes

Support for sensitive Resultset : yes

Support for insensitive Resultset : no

Support for updatable Resultset : yes

#### oracle thin / OCI / Connector-J driver for MySQL / PostgreSQL thin Driver

Support for Non-scrollable Resultset : yes

Support for scrollable Resultset : yes

Support for Readonly Resultset : yes

Support for sensitive Resultset : no

Support for insensitive Resultset : yes

Support for updatable Resultset : no

\* For example application on Updatable ResultSet object refer application 18 of page no. 30.

### Rowsets:

\* JDBC Resultset objects are not serializable objects so we can not send those objects over the network.

\* To solve this problem we can use Rowsets as alternate for Resultsets.

\* JDBC Rowsets are serializable objects by default.

\* Rowset object means it is the object of a Java class that implements javax.sql.RowSet interface. This interface is subinterface of java.sql.ResultSet interface.

\* we can develop Rowsets in three modes.

(1) As Cached Rowset → It is a disconnected Rowset.

→ It is like insensitive ResultSet object.

(2) As JDBC Rowset → It is a connected Rowset.

→ It is like sensitive ResultSet object.

(3) As web Rowset → It is a connected Rowset.

→ Allows to read data from XML files.

\* Oracle corporation gives support for two types of Rowsets (cached Rowset, JDBC Rowset). For this it supplies oracle.jdbc.rowset.OracleCachedRowset class and oracle.jdbc.rowset.OracleJdbcRowset class. These two classes are available in ojdbc14.jar file of oracle 10g, whereas in oracle 9i these two classes are available in ocrsg12.jar file and for this jar file oracle for ojdbc14.jar file is dependent jar file.

\* Working with Rowsets is always an easy process because it allows getter and setter methods based programming.

\* For example application on Rowsets refer application 19 and 20 of page nos. 30 and 31.

## Advantages with Rowsets

- (1) Gives simplified programming (allows single object based setter and getter methods oriented programming).
- (2) Rowsets are serializable objects so they can be sent over the network.

## Disadvantages

- (1) very few drivers are supporting Rowsets.
- (2) Non-select operations on database table using Rowsets is not possible.

NOTE: All Collection framework data structures are serializable objects by default. Since very few JDBC drivers are giving support for Rowsets the programmers copy ResultSet object records to collection framework data structure (like ArrayList) and sends that collection framework data-structure over the network.

\* If classes of certain jar file uses the classes and interfaces of certain other jar files then the other jar files are called as dependent jar files to the first main jar file.

Ex: The classes of ocrs12.jar file internally uses the classes of jdbc14.jar file so we can say jdbc14.jar file dependent jar file to ocrs12.jar.

## Working with Date Values

\* While dealing with DOB, Date of joining, Bill purchase date and etc values we need to see JDBC persistence logic performing insert, update, delete and select operations on date values.

\* Don't insert date values in database table columns as String values because we can't apply the date related SQL functions on those date values.

\* It is always recommended to insert the date values in "date" data type columns of database table.

\* Different database softwares support different patterns of date values like Oracle supports → dd-mm-yy pattern (ex: 20-jan-99)

mysql supports → yyyy-mm-dd pattern (ex: 1999-10-20)

\* once java.sql.Date class object is given to jdbc driver representing Date value then the JDBC driver inserts that date value in the database table column in the pattern that is supported by underlying database software.

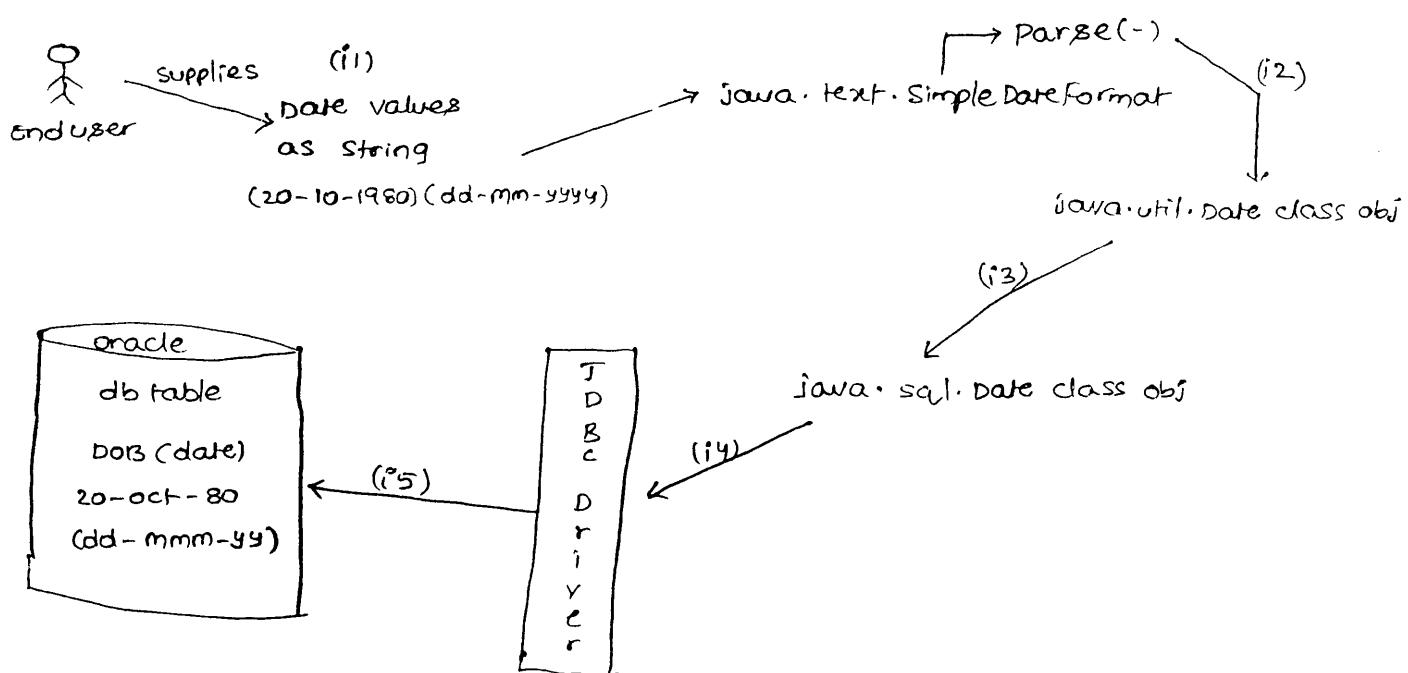
### wrong database table design

<u>name</u>	<u>dob</u>
(vc2)	(vc2)
raja	20-10-1982
ramesh	30-10-1987

### Good database table design

<u>name</u>	<u>dob</u>
(rc2)	(date)
raja	20-oct-1982
ramesh	30-oct-1987

Database software and JDBC driver independent logic to insert date value in date datatype column of database table in the pattern that is supported by underlying database software.



with respect to diagram

i1 - End user supply string date value to application

i2 - The parse() method of simple date format class converts string date value to java.util.Date class object.

i3 - Application converts java.util.Date class object to java.sql.Date class object.

i4 - Application gives this java.sql.Date class object to jdbc driver using PreparedStatement object

i5 - This jdbc driver inserts the date value to database table column in the pattern it is supported by database software.

What is the difference between java.util.Date class and java.sql.Date class?

#### java.util.Date

\* This class object represents ordinary date values, but it can't represent db table column date values.

\* JDBC driver can't recognize the date value of this class object.

\* java.sql.Date is the sub class of java.util.Date.

#### java.sql.Date

\* Represents db table stat column date values.

\* JDBC driver can recognize the date value of this class object.

//Converting String date value to java.util.Date class object

```
String s1 = "20-10-99"; // dd-MM-yy
```

```
SimpleDateFormat sdf1 = new SimpleDateFormat ("dd-MM-yy");
```

```
java.util.Date udl = sdf1.parse(s1);
```

```
sop("util date is :" + udl.toString());
```

The date pattern of input  
date value

//Converting java.util.Date class obj to java.sql.Date class object

```
long ms = udl.getTime(); ①
```

```
java.sql.Date sqd1 = new java.sql.Date(ms);
```

```
sop("sql Date is :" + sqd1.toString());
```

① - gives no. of milliseconds that are elapsed between january 1st 1970 00:00 hours (Epoch standard) and the date and time represented by udl object (java.util.Date class object).

//Converting string date value directly to java.sql.Date class object

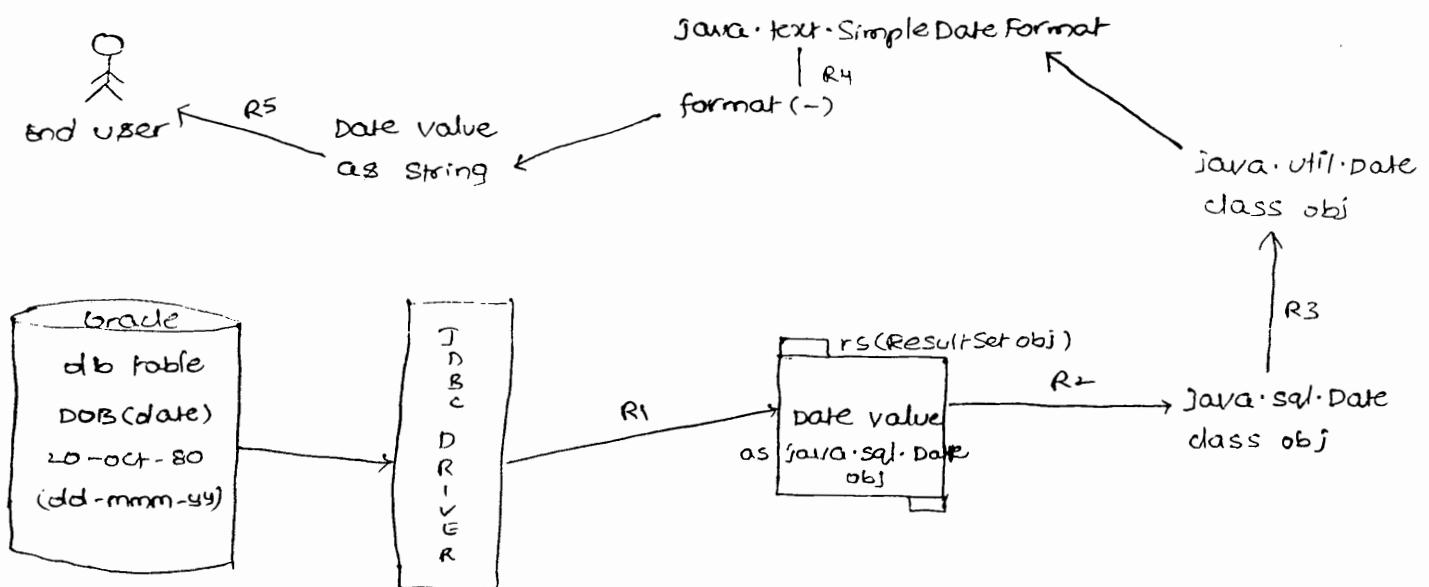
//Note: the String value must be there in yyyy-MM-dd pattern)

```
String s2 = "1982-10-20"; //yyyy-MM-dd
```

```
java.sql.Date sqd2 = java.sql.Date.valueOf(s2);
```

```
sop("the sql date is:" + sqd2.toString());
```

The db s/w independent and jdbc driver independent procedure to retrieve dates from database table column



R1 - Application executes select query on database table and gets jdbc resultset object having date value

R2 - Application retrieves date value from Resultset object as java.sql.Date class object.

R3 - Application converts java.sql.Date class object to java.util.Date class object.

R4 - The format(-) method of SimpleDateFormat class converts java.util.Date class object to String date value in the pattern that is expected by end user.

R5 - Application displays string date value for end user.

```
// Converting java.sql.Date class obj to java.util.Date class obj  
java.util.Date ud2 = (java.util.Date) sqd2;  
System.out.println("The util date value is: " + ud2.toString());  
  
// Converting java.util.Date class object to string date value  
SimpleDateFormat sdf2 = new SimpleDateFormat("MMM-yy-dd");  
String s3 = sdf2.format(ud2);  
System.out.println("String date value is: " + s3);
```

↓  
Expected date pattern  
of o/p string date value

### Example application to insert and retrieve data values

#### DB Table (oracle)

```
create table person_tab (no number, name varchar2(20), DOB date,  
                        DOJ date);
```

#### DB Table (mysql)

```
create table person_tab (no int(5), name varchar(20), DOB date,  
                        DOJ date);
```

#### // DateInsert.java

```
import java.util.*;  
import java.sql.*;  
import java.text.*;  
public class DateInsert  
{  
    public static void main(String args[]) throws Exception  
    {  
        // read input values from key board  
        Scanner sc = new Scanner(System.in);  
        System.out.println("Enter person number:");  
        int no = sc.nextInt();  
        System.out.println("Enter person name:");  
        String name = sc.next();  
        System.out.println("Enter person date of birth");  
        String pDOB = sc.next();  
        System.out.println("Enter person date of joining");  
        String pDOJ = sc.next();
```

```

//convert string date values to java.sql.Date class object.

//for date of birth
SimpleDateFormat sdf1 = new SimpleDateFormat("dd-mm-yyyy");
java.util.Date udob = sdf1.parse(pdob);
java.sql.Date sqdob = new java.sql.Date(udob.getTime());

//for date of joining (yyyy-MM-dd)
java.sql.Date sqdoj = java.sql.Date.valueOf(pdoj);

//create JDBC connection object
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@local
host:1521:ord", "scott", "tiger");

//create Prepared statement object pointing to insert query
PreparedStatement ps = con.prepareStatement("insert into person_tab
values (?, ?, ?, ?, ?)");

//set values to the parameters of query
ps.setInt(1, no);
ps.setString(2, name);
ps.setDate(3, sqdob);
ps.setDate(4, sqdoj);

//execute the query
int res = ps.executeUpdate();

//processing the result
if (res == 0)
    sop("record not inserted");
else
    sop("record is inserted");

//close JDBC objects
ps.close();
con.close();

}//main

}//class
//javac DateInsert.java
//Java DateInsert

```

\* using simple statement object we cannot insert date, large object values (files in database table). But we can do this work by using PreparedStatement object.

### Application

```
//DateRetrive.java
import java.sql.*;
import java.util.*;
import java.text.*;
public class DateRetrive
{
    public static void main (String args[])
        throws Exception
    {
        //create jdbc connection object
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:
                                         localhost:1521:orcl","scott","tiger");
        //create jdbc statement object
        Statement st = con.createStatement();
        // execute the query
        ResultSet rs = st.executeQuery("select * from person_tab");
        // process the result set
        while (rs.next())
        {
            int no = rs.getInt(1);
            String name = rs.getString(2);
            java.sql.Date sqdob = rs.getDate(3);
            java.sql.Date sqdobj = rs.getDate(4);
            //Convert java.sql.Date class object to java.util.Date class
            //object
            java.util.Date udob = (java.util.Date) sqdob;
            java.util.Date udoj = (java.util.Date) sqdobj;
            //Convert java.util.Date class objects to String class values
            SimpleDateFormat sdf1 = new SimpleDateFormat("MMM-yy-dd");
            String pdob = sdf1.format(udob);
            SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy-mm-dd");
            String pdoj = sdf2.format(udoj);
```

```

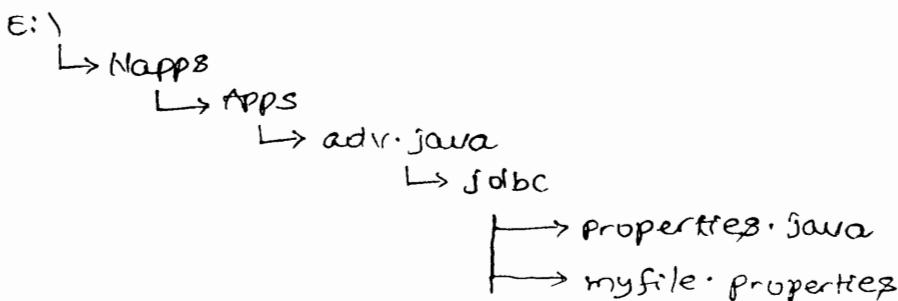
        sop("No " + name + " " + pob + " " + Pdoi);
    } //while
    //close jdbc objects
    rs.close();
    con.close();
    st.close();
} //main
} //class.

```

\* Don't hard code any values in your java application that are changeable in the future, pass them to application from outside the application by taking the support of properties file.

\* To make jdbc code of java application as flexible code to modify collect the following details from outside the application by taking the support of properties file the following

- \* JDBC class name
- \* URL
- \* Database username
- \* database password
- \* sql query



### Properties file

```

# jdbc details
my.driver = oracle.jdbc.driver.OracleDriver
my.dburl = jdbc:oracle:thin:@localhost:1521:orcl
my.dbuser = scott
my.dbpwd = tiger
my.sql = select * from student
→ save myfile.properties

```

```
//PropTest.java
import java.sql.*;
import java.util.*;
import
public class PropTest
{
    public static void main(String args[])
        throws Exception
    {
        //Locate text properties file
        FileInputStream fis = new FileInputStream("myfile.properties");
        //create an empty object of java.util.Properties class
        Properties p = new Properties();
        //load the content of myfile.properties into java.util.Properties
        //class
        p.load(fis);
        //read values from java.util.Properties class of object (p)
        String dname = p.getProperty("my.driver");
        String url = p.getProperty("my.dburl");
        String user = p.getProperty("my.dbuser");
        String pwd = p.getProperty("my.dbpwd");
        String query = p.getProperty("my.sql");
        //create jdbc connection object
        Class.forName(dname);
        Connection con = DriverManager.getConnection(url, user, pwd);
        //create statement object
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(query);
        while(rs.next())
        {
            System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3) + " "
                + rs.getString(4));
        }
        //close jdbc objects
        rs.close();
        con.close();
        st.close();
    }
}
```

\* In database table files are called as large objects.

\* There are two types of large objects

(1) BLOB → Ex: image files, avi files

(2) CLOB → Ex: text files, rich text files, ms word documents

+ matrimony, job portal applications we must deal with these large objects.

\* To insert LOB values to database table use JDBC Prepared Statement

object and take column data types has BLOB or CLOB.

standard procedure to insert LOB values to database table column

1. Create InputStream object pointing to file

```
FileInputStream fis = new FileInputStream("c:\abc\ash.gif");
```

2. Create PreparedStatement object pointing insert<sup>query</sup> based on JDBC Connection object  
PreparedStatement ps = con.prepareStatement("insert into EmpAll values(?, ?, ?, ?);")

3. Set values to parameters of query

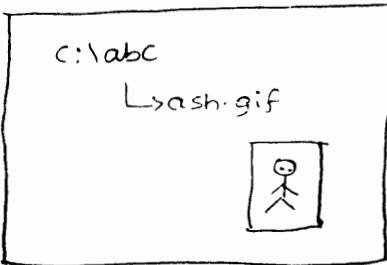
```
ps.setInt(1, 123); ps.setString(2, "ashb"); ps.setInt(3, 50000);
```

```
ps.setBinaryStream(4, fis.length);
```

4. Execute the query

```
ps.executeUpdate();
```

File location



oracle db s/w  
EmpAll (db table)

eno	ename	bsal	ephoto
(n)	(c2)	(n)	(BLOB)
123	a.shb	50000	

\* All files and directories are belonging to various drives together is called as Filesystem.

\* To binary large object values to query we can use setBinaryStream(sp) or setBlob(bp) method.

\* Similarly to set character large object values to SQL query we can use

`setCharacterStream()` or `setClob()` methods.

\* For example application on working with large object refer application 23 and 24 of hand book.

standard procedure to follow to retrieve large object values from database table column

Step(1): Execute select query on database table and get JDBC ResultSet object.

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

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

Step(2): Get access ephoto value from db table

```
rs.next();
```

```
FileInputStream fis = rs.getBinaryStream("ephot");
```

↑

Input Stream object pointing photo db table column through  
ResultSet object

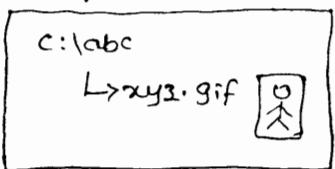
Step(3): Create FileOutputStream object pointing to destination file (xyz.gif)

```
FileOutputStream fos = new FileOutputStream("c:\abc\xyz.gif");
```

Step(4): Write db table column photo to the destination file (xyz.gif)  
using both input, output streams objects.

```
int bytesRead = 0;  
byte[] buffer = new byte[4096];  
while (bytesRead = fis.read(buffer) != -1)  
{  
    fos.write(buffer, 0, bytesRead);  
}
```

computer file



oracle db table

empAll

eno ename bsal ephot  
123 ashb 50000



\* Buffer is temporary memory to store temporary data.

\* The process of storing data in buffer before sending to final destination is called as buffering.

\* While transferring source file data to destination file if buffer support

is taken, we can reduce no. of read and write operations on source and destination files respectively.

\* In the above code the bytesRead variable keeps track of the no. of bytes that are stored into buffer each time by reading the content from source file (image of ResultSet object) so that we can use that data value while writing buffer content to destination file.

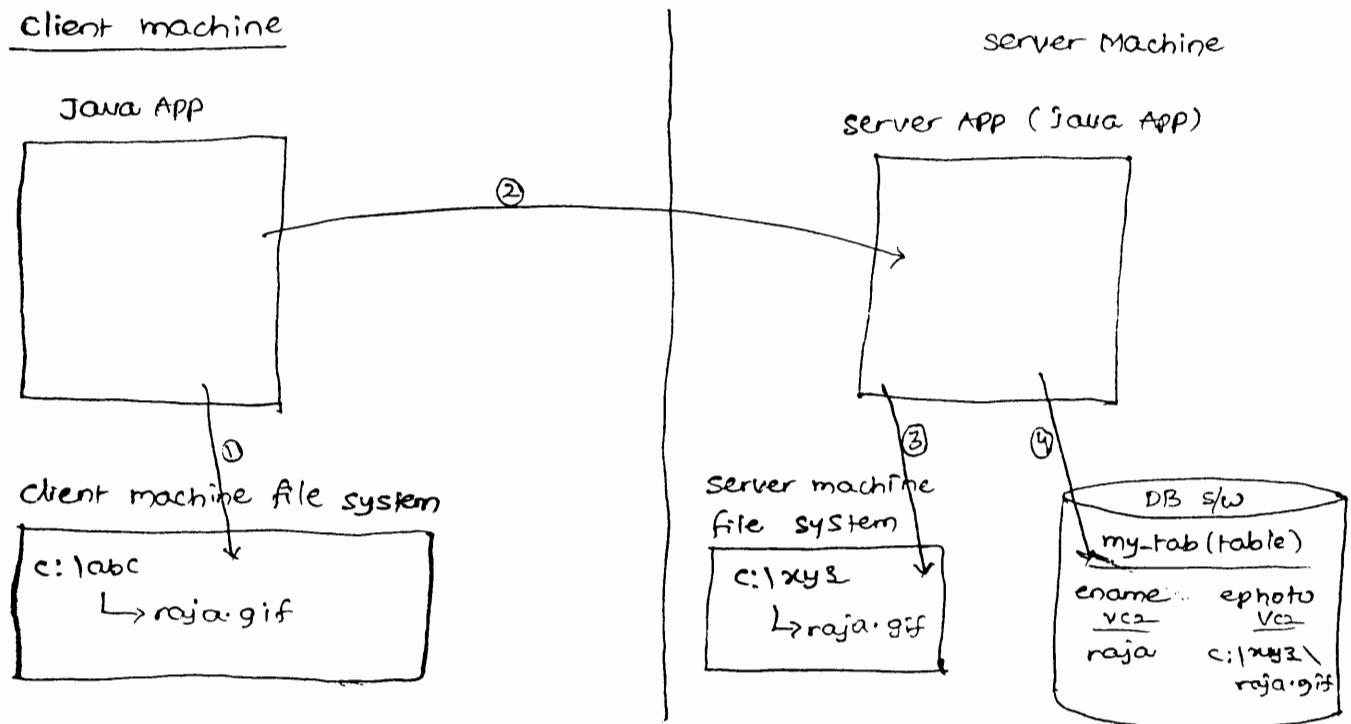
\* To read BLOB value from ResultSet object use rs.getBinaryStream() or rs.getBlob() method.

\* To read CLOB value from ResultSet object use either rs.getCharacterStream() or rs.getCharacterStream() method.

\* In real time projects no one prefers to store large object value like image files, text files etc...

\* Directly in database table columns because they may degrade performance of database software, by allocating memory upto 4GB.

+ In industry level projects instead of storing LOB files directly as VB table column values they store these files in computer file system/directories of hard disk drives and they will store the paths of these files as string values in String data type table columns.



\* The process of sending file from client machine to server machine is called as file uploading and reverse is called as file downloading.

Sample Scenario to know how realtime projects manages large objects (towards insertion)

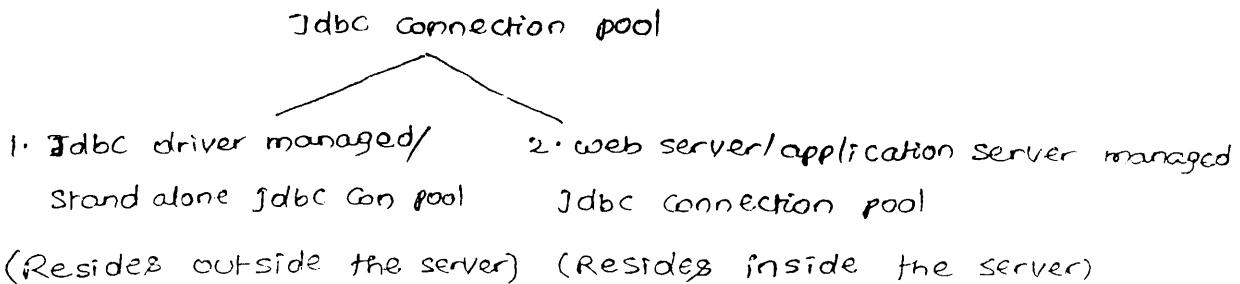
- (1) client application locates and selects the file from client machine file system.
- (2) client application sends the file to server application through file uploading process.
- (3) Server application writes/saves the received file to the file system of server machine.
- (4) Server application writes the path of the file on server machine as db table column value.

\* JDBC connection pool is a factory that contains set of ready available JDBC connection objects before actually being used.

\* The advantage with JDBC connection pool is we can use minimum no. of Connection objects to give service to maximum clients.

\* There are two types of JDBC connection pools:

- (1) JDBC Driver managed/stand alone JDBC Connection
- (2) web server/application server managed JDBC connection



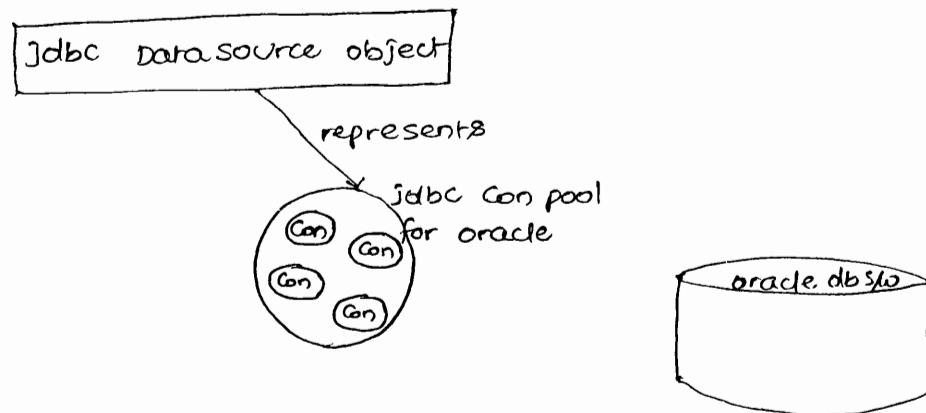
\* While developing small scale applications like stand alone, desktop application we can use driver managed JDBC connection pool.

\* While developing large scale applications like web sites and enterprise applications we can use server managed JDBC connection pool.

\* All JDBC connection objects in a JDBC connection pool represents connectivity to same database software.

\* JDBC connection pool for oracle means all JDBC connection objects in that connection pool represents connectivity with oracle database software.

- \* oracle thin driver/ oci driver gives one built-in jdbc connection pool for oracle.
- \* JDBC data <sup>source</sup> object represents jdbc connection pool so our application should always get a JDBC connection object from connection pool through data source object



\* JDBC data source object means it is the object of a java class that implements javax.sql.DataSource (I).

\* The JDBC connection object that is created by the programmer manually is called as direct JDBC Connection object.

\* The JDBC connection object that is collected from JDBC connection pool is called as pooled JDBC connection object.

#### Example application on oracle thin driver managed JDBC Connection pool

```
//Connectionpooltest.java
import java.sql.*;
import javax.sql.*; // for pooledConnection(I)
import oracle.jdbc.pool.*; // (for oracle Connection pool DataSource(I))

public class ConnPoolTest
{
    public static void main(String args[]) throws Exception
    {
        //create JDBC data source object representing empty driver managed
        //connection pool of oracle

        OracleConnectionPoolDataSource ds = new OracleConnectionPoolData
                                         Source();
        //gives details to create JDBC connection objects in the connection pool
        ds.setDriverType("thin");
    }
}
```

```

        ds.setServerName("localhost");
        ds.setPortNumber(1521);
        ds.setServiceName("ord");
        ds.setUser("scott");
        ds.setPassword("tiger");

    //get & Access to jdbc connection pool for oracle (this connection
        //pool will be created based on the above details)

    PooledConnection con = pcon.getConnection();
    //write jdbc based persistiance logic

    Statement st = con.createStatement();

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

    while (rs.next())
    {
        System.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
    }

    //close jdbc objects

    rs.close();
    st.close();
    con.close(); // releases connection object back to jdbc con pool.

} //main
} //class
//add ojdbc14.jar file to classpath and then execute.

```

- \* Scrollable Resultset, sensitive Resultset, updatable Resultset features are given from jdbc 2.0 onwards.
- \* Driver manager Connection pool save point Connections are given from jdbc 3.0 onwards.
- \* Auto loading of driver class annotations based jdbc programming features are given from jdbc 4.x onwards.
- \* To work with Driver managed jdbc connection pool there is no necessity of dealing with type3 protocol. While working with server manager connection pool we must deal with type3 protocol.
- \* Save point is a point within the current transaction that can be referenced from the Connection.rollback method.

- \* When a transaction is rolled back to a save point point all the changes made after that savepoint point will be under.
- \* After executing 10 queries if you want to commit first 5 query results and if you want to rollback last 5 query results then create savepoint after first 5 queries and rollback upto that savepoint position.
- \* There are two types of savepoints that can be created in jdbc environment.
  - (1) named savepoint
  - (2) unnamed savepoint

NOTE: For unnamed savepoint the db s/w automatically generates an identity value.

- \* In jdbc queries based transaction save point based rollbacks are possible but savepoint based commits are not possible.

```
//SavepointTest.java
import java.sql.*;
public class SavepointTest
{
    public static void main(String args[])
        throws Exception
    {
        //create jdbc connection object
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                                    :1521:orcl", "scott", "tiger");
        Statement st = con.createStatement();
        //Begin Tx
        con.setAutoCommit(false);
        //query1
        st.executeUpdate("insert into student values (671, 'xyz', 'hydi')");
        //set a named savepoint to connection object
        Savepoint svpt = con.setSavepoint("mysp"); //mysp is savepoint
                                                       name
        //query2 in savepoint area
        st.executeUpdate("update student set name='mahesh2' where
                        sno=345");
```

```

    //rollback upto savepoint    con.rollback(svpt);
    //con.commit(svpt); //error
    //Commit Tr
    con.commit();
    //record insertion will be committed (query1) but record
    //updation (query2) will be rolledback because query2 is
    //there in save point area

} //main
} //class
//> javac SavepointTest.java
//> java SavepointTest

```

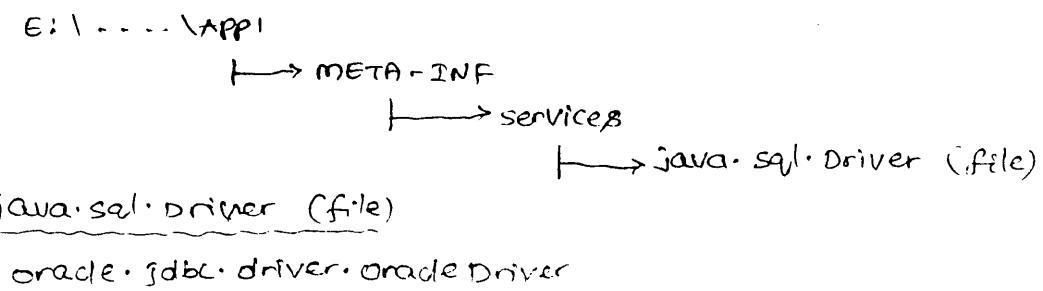
\* To work with JDBC 3.0 features we need Jdk 1.5 or higher version  
 similarly to work with JDBC 4.0 features we need Jdk 1.6 or higher version setup.

\* The auto driver class loading feature of JDBC 4.0 can load the make the application loading Driver class automatically in one of the following two situations.

- (1) If Driver class is specified in META-INF/java.sql.Driver file of application directory related work directory.
- (2) If jar file that represents JDBC driver contains java.sql.Driver file having JDBC driver class name.

\* JDBC 4.0 does not contain java.sql.Driver file so you must arrange that file explicitly as shown below to make application loading oracle thin Driver class automatically as shown below.

Step(1): create java.sql.Driver file



Step(2): Develop java application as shown below

```
//SelectTest.java
import java.sql.*;
public class SelectTest
{
    public static void main (String args[])
        throws Exception
    {
        //do notice that there is no class.forName(-)
        Connection con = DriverManager.getConnection ("oracle.jdbc;oracle:
thin:@localhost:1521:orcl", "Scott", "Tiger");

        if (con != null)
            System.out.println ("Connection is established");
        if (con == null)
            System.out.println ("Connection is not established");
    }
}
```

javac SelectTest.java

java SelectTest

Example(2):

\* The postgresql related thin Driver jar file is having `java.sql.Driver` file having the driver class name. By just adding this jar file to class path we can make our java applications loading jdbc driver class automatically.

E:\---\APP2

→ SelectTest.java

```
//SelectTest.java
import java.sql.*;
public class SelectTest
{
    public static void main (String args[])
        throws Exception
    {
        //do notice that there is no class.forName(-)
        Connection con = DriverManager.getConnection ("jdbc :postgresql:
postgres", "postgres", "postgres");
    }
}
```

```
if (con != null)
    sop ("Connection established");
else
    sop ("connection not established");

//main
//class
//>javac SelectTest.java
//>java SelectTest.java
```

\*Before executing above example add postgresql-8.4-701.jdbc4.jar file to class path.

———— \* \* \* ——

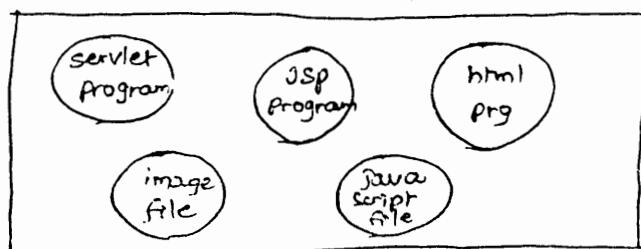
## Servlets

- \* The logics and data of stand alone, desktop applications ~~is~~ <sup>are</sup> specific to that computer where these applications are running.
- \* The logics and data of client-server applications, two-tier applications like JDBC applications are specific to one network where they are running. In the above said two tier, client-server applications the server allows only known clients (recognized clients).
- \* To provide global visibility and accessibility to the logics and data of the application use the internet environment based websites or web applications. These applications allow both known and unknown clients having 24x7 access to the resources and logic data.
- \* When website is under development / testing in a software company then it is called as web application. Once web application is hosted on to the internet network by purchasing domain name (www.citibank.com) or host name and space in the internet network is called as website.
- \* A web application is a collection of web resource programs which can generate web pages. There are two types of web resource programs based on the web page they generate.
  - 1) Static web resource programs → generates static webpages (Ex: HTML)
  - 2) Dynamic web resource programs → generates dynamic webpages  
Ex: (Servlet program, JSP programs, ASP programs, PHP programs)

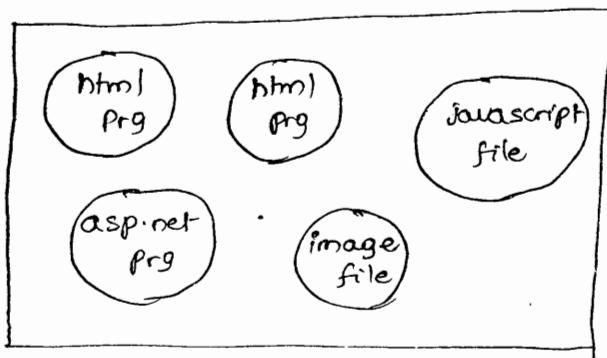
↓  
Personal Home Page for Hypertext processing

- \* Static web page contains fixed content forever. The content of dynamic web page changes based on the input values of the request or time of the request generation.

Java based web application



## .net Based web Application



- \* The web resource programs like image file, javascript file can not generate web pages directly but they support other web resource programs to generate web pages.
- \* Stand alone applications, desktop applications and client-server applications will be executed by programmer manually. But the web resource programs of web application must be executed dynamically when they are requested by clients (Browser windows) so we can't think about manual execution for web resource programs. We can use a special software called web server software to execute the web resource programs dynamically when they are requested by clients.
- \* A web server software can also manage, execute multiple web applications simultaneously or parallelly.

### Examples of web server / HTTP server softwares

Tomcat

Resin

JWS (Java web server)

PWS (Personal web server)

IIS (Internet Information Server)

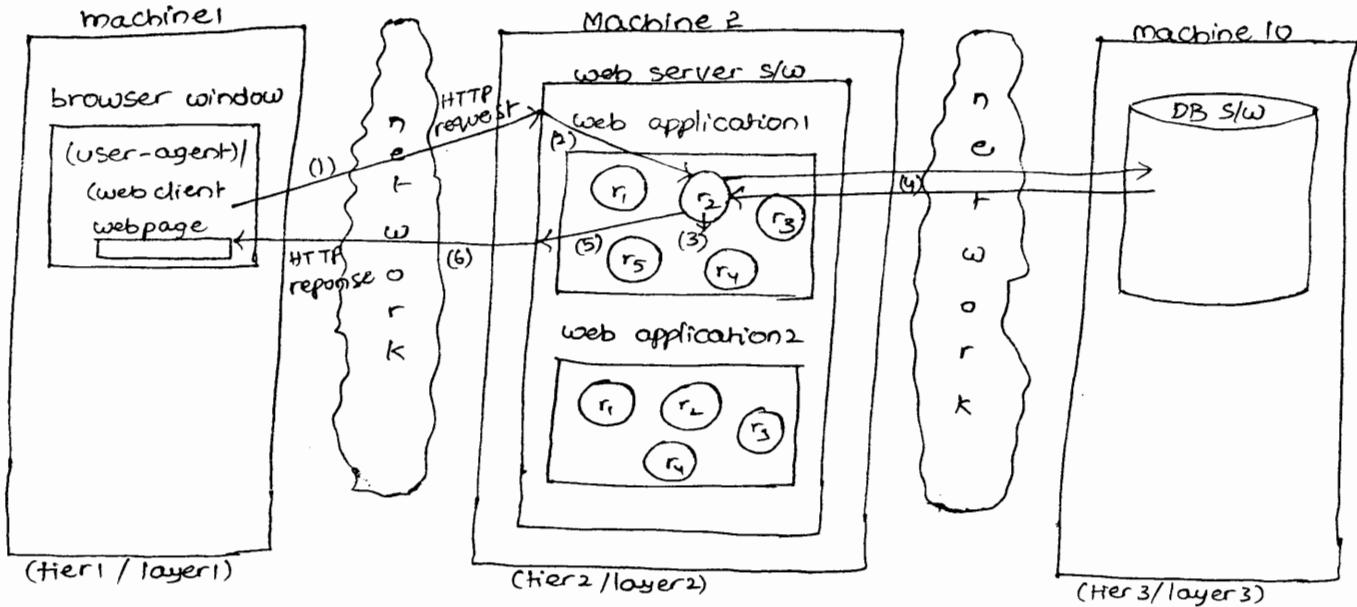
and etc...

- + To develop and execute JDBC application we need the following setup.  
JDK s/w, JDBC driver, DB s/w.
- + Similarly to develop and execute web application we need the following setup.  
Browser s/w, Web server s/w, DB s/w, Web technologies → to develop web resource programs

\* web server s/w used to manage and execute web application.

\* browser s/w used to request to web application.

\* java based web servers internally uses jdk s/w  
setup required to develop and execute web application



\* Each browser window acts as one client to web application.

\* A web application can be developed as two tier application without database software or three tier application with database software.

\* with respect to diagram

(1) Browser window generates http request to web application.

(2) The web server traps and takes the request and passes the request to appropriate web resource program.

(3) The web resource program process the request.

(4) Web resource program interacts with database software if necessary.

(5) The web resource program generated result goes to web server.

(6) Web server sends these results to browser window as HTTP response in the form of web page.

\* In the above diagram r<sub>1</sub>, r<sub>2</sub>... are web resource programs.

\* Based on the place of executing where web resource programs execute there are two types of web resource programs.

(1) server side web resource programs (they execute in the web server).

Ex: servlet prg, JSP prg and etc.

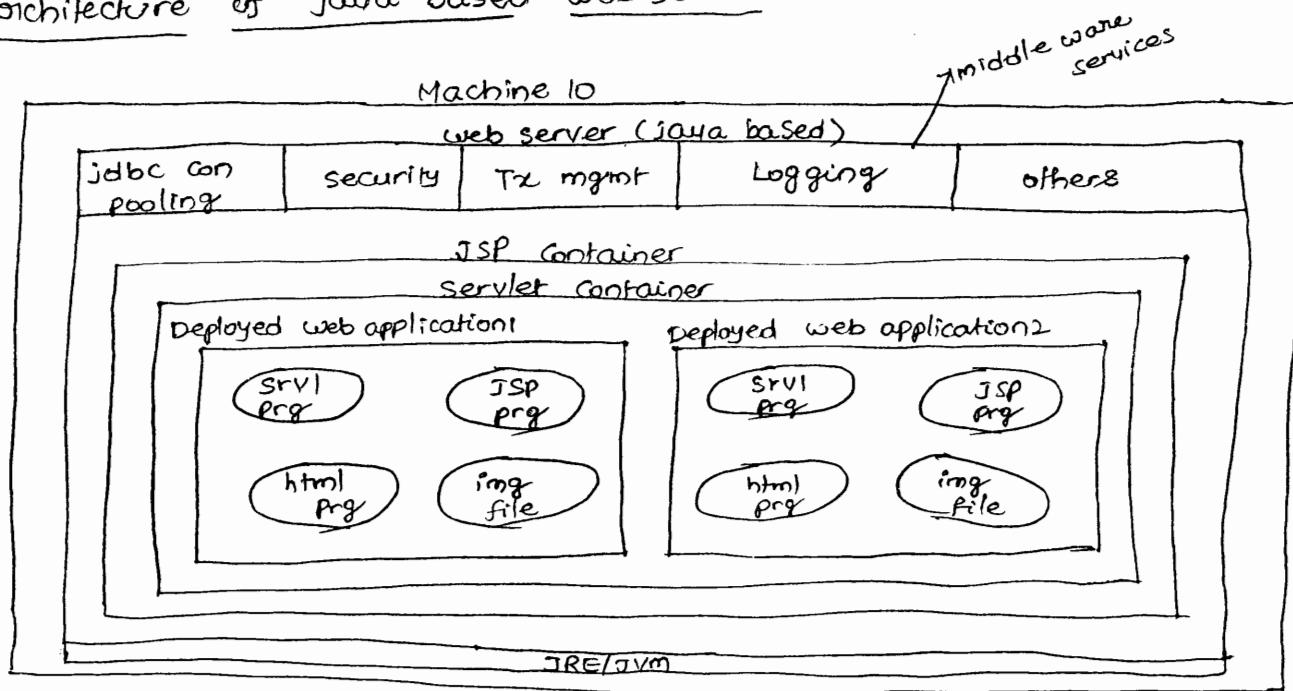
- (2) client side web resource program → They come to browser window from the web application placed in web server for execution.  
Ex: HTML program, Javascript program, Ajax programs and etc..
- \* Decide whether web resource program is client side or server side based on the place where it executes. not based on the place where it resides.
- \* A web application looks like thin client-fat server application.
- \* The process of keeping web application in web server is technically called as deployment and reverse process is called as undeployment.

### Responsibilities of web server

- 1) Listens to client request Continuously (HTTP request)
  - 2) Traps and takes client generated HTTP request.
  - 3) Passes the HTTP request to an appropriate web resource program of web application. (deployed web application)
  - 4) Provides container software to execute server side programs (web resource programs)
  - 5) Gathers output generated by web resource programs.
  - 6) Passes output of web resource programs to browser window as http response in the form of web pages.
  - 7) Provides environment to deploy manage and to undeploy the web application.
  - 8) Gives middle ware services and etc...
- \* A Container is a software or software application that can manage the whole life cycle (object birth to death) of given resource. (like java class).
  - \* A file or program is called as resource of the application.
  - \* Servlet Container takes care of servlet program lifecycle.
  - \* JSP Container takes care of JSP program lifecycle.
  - \* Applet Container (Applet viewer) takes care of applet program lifecycle.
  - \* Servlet Container, JSP containers are part of web servers (java based).  
→ Container is like a aquarium who can take care of the whole lifecycle of given resources called fishes.

\* programmer is not responsible to develop containers and web servers but he is responsible to use them to execute web applications.

### Architecture of java based web server



\* Once web server is started one daemon process will be started to listen to client's continuously and to trap and take the client generated HTTP request.

\* The process that runs continuously is called as <sup>daemon</sup> ~~daemon~~ process.

\* Every java application contains two default threads.

(1) main thread

(2) Garbage collector thread (daemon thread).

\* JSP Container is the enhancement of servlet Container and both Containers use JRE/JVM supplied by web server.

\* middle ware services are configurable additional services on the application to make applications executing perfectly in all the situations.

\* security middle ware service protects the application from unauthorized and unauthenticated users.

\* Transaction management service executes the logics by applying do everything or nothing principle.

\* JDBC connection pool supplies set of ready available JDBC connection objects.

- \* Logging service keeps track of the application execution process through Config conformation statements / messages / log messages.
- \* middleware services are not minimum logics of application development. They are additional or optional services to apply on applications.
- \* A web application can be there with or without middleware services.
- \* web server automatically activates servlet Container and JSP Container to execute servlet, JSP programs when they are requested by clients.
- \* the client side web resource programs of web application goes to browser window for execution whereas the server side programs will be executed by using the containers of web server.
- \* In web application executing environment browser software is called as client software and web server software is called as server software.

#### Examples of java based web servers

Tomcat → from Apache Foundation  
 Resin → from Resin soft  
 JWS → from Sun microsystem

\* HTTP is a application level protocol define in set of rules to get communication between browser window and web server. The text that allows sequential reading is called as normal text.

Ex: notepad text.

- \* The text that allows non-sequential reading through hyperlinks is called as hyper text.
- Ex: any web page content.
- \* Protocol HTTP is useful to transfer hyper text between browser window and web server, web server and browser window.

#### Different browser softwares

IE	— Microsoft
Netscape Navigator	— from Netscape
Fire Fox	— from Mozilla
Ice Java	— from RedHat

chrome — from Google

opera — from opera soft

## Different client side Technologies (for developing client side web resource programs)

html → from W3C

java script → from netscape

VB Script → from Microsoft

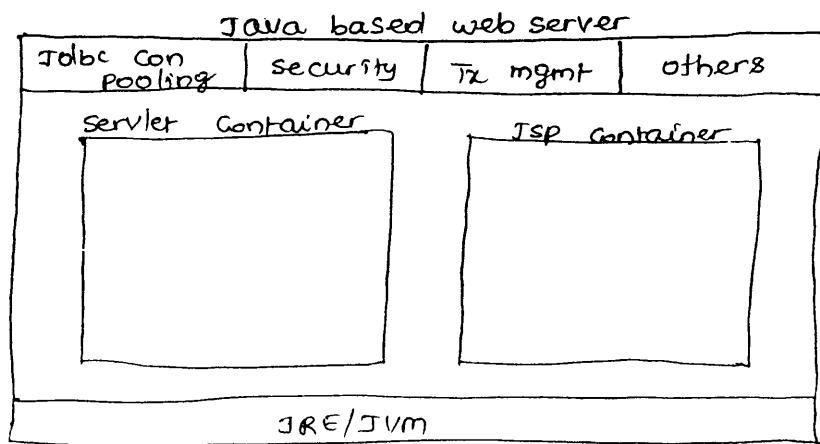
AJAX → from Adaptive path

NOTE: Java script is no way related with the programming language java.

\* The code that can not be executed independently and which must be embedded in other technology based program for execution is called as script code.

\* Java script code can not be executed independently and it must be executed by embedding with html code so java script is called as scripting language.

Another diagram that shows Java based web server architecture



Jsp container is developed based servlet container

\* when java web application is deployed in java web server then the servlet programs will be of that web application will be executed by servlet container and Jsp programs will be executed by Jsp container.

\* Directly or indirectly every java application (or) java resource (or) java component uses JVM during execution.

Ex: appletviewer executes our applet program but this applet viewer internally uses (or) take support from JVM.

Servlet Container executes servlet program but this servlet container internally uses JVM support

### Different server side Technologies (to develop server side web resource programs)

Servlet → from sun microsystems → java based (2)

JSP → from sun microsystems → java based (3)

ASP → from microsoft → (non-java based)

ASP.NET → from microsoft → (non-java based) (4)

PHP → from Apache → (non-java based) (5)

COLDfusion → from Adobe → (non-java based)

SSJS → from Netscape → (non-Java based)

(Server side  
Java script)

### Different web Server softwares

Tomcat → from Apache → Java based (1)

Resin → from Resin soft → java based (2)

JWS → from Sun microsystems → java based

Jetty → from Adobe → Java based

IIS → from microsoft → Non-Java based (3)

PWS → from microsoft → (Non-Java based)

AWS → from Apache → Non-Java based (4) (for PHP programs)

(Apache web server)

} for servlet, JSP  
Programs

} for ASP, ASP.NET  
Programs

Application Server = Web server + EJB Container + more middleware services

NOTE: EJB Container is required to execute EJB Components.

→ A reusable Java class object is called as Component.

Application server software is enhancement of web server software.

→ Every application server software can act as web server.

### Different application server softwares

Weblogic → from BEA systems (Oracle Corporation) (1)

Websphere → from IBM (3)

Iboss → from Apache / Red Hat (4)

GlassFish → from sun microsystems (2)

OracleAS → from Oracle Corporation

(oracle log  
Application server)

Jrun → from macromedia (Adobe)

\* All application server softwares are java based softwares.

#### Different database softwares

oracle → oracle corporation (1)

mysql → from devx (3)

SQLServer → from microsoft (2)

DB 2 → from IBM

PostgreSQL → from university of California (4)

\* We can use client side technologies like HTML, JavaScript along with any vendor supplied server side technologies in web application development.

\* To develop & execute Java based web application

a) choose any browser software

b) choose one or more client side technologies like HTML, JavaScript

c) choose servlet, JSP as server side technologies.

d) choose any java based web server software or application server software.

e) choose any data base software

\* To develop and execute .NET based web application.

a), b), e) are same as above

c) choose ASP.NET as server side technology

d) choose Microsoft supplied IIS, PWS server.

\* To develop PHP based web applications

a), b), e) are same as above

c) choose PHP as server side technology.

d) use AWS as web server.

\* To develop medium scale and large scale web applications use Java environment.

\* To develop small and medium scale web applications use .net environment.

\* To develop small scale web applications use PHP environment.

## Tomcat

Type : Java based web server

version : Tomcat 6.0 (Compatible with JDK 1.6)

Tomcat 7.0 (Compatible with JDK 1.6/1.7)

Tomcat 5.0 (Compatible with JDK 1.5)

Vendor : Apache foundation

open source software (the source code will be exposed)

default port no: 8080 (changeable)

To download software: www.apache.org

for docs : www.apache.org

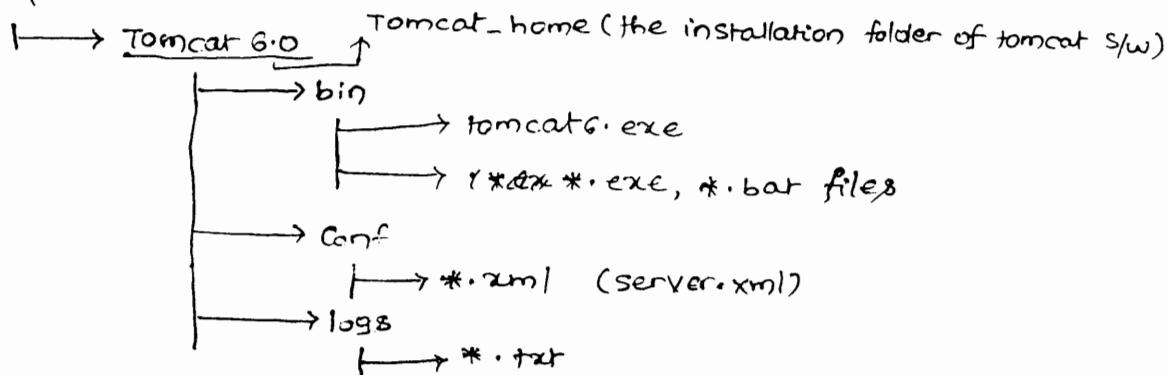
for faqs : www.forum.apache.org

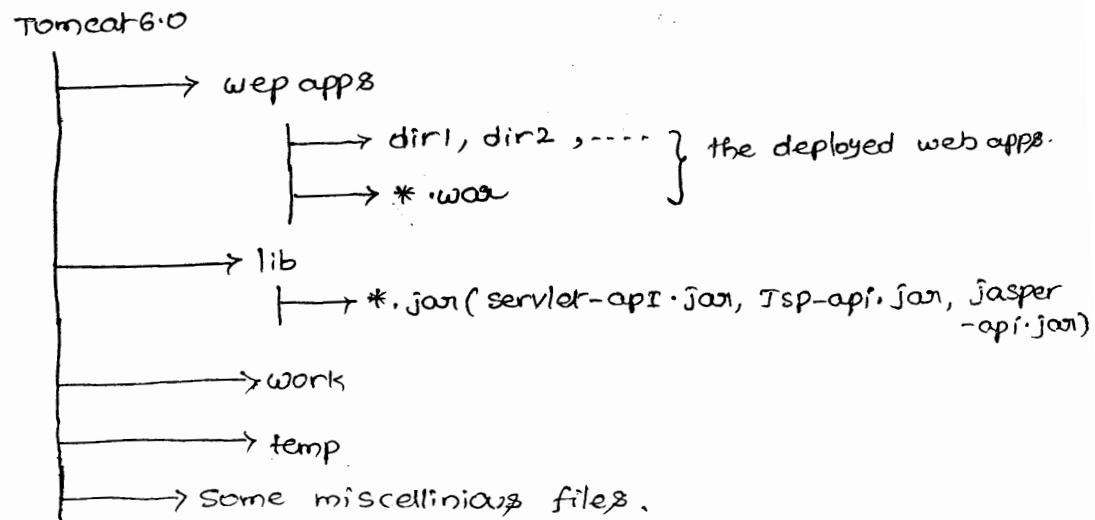
\* To install Tomcat 6.0 software use Apache → Tomcat - 6.0.26 set up file and you can change the default port no, default username and password details during installation. If not changed default port no is 8080, default username is "admin", default password is "admin".

\* Every software installed in a computer will reside in a logical position called software port and every software port will be identified with its port no. (Communication end point)

After installing tomcat you will get the following folders and files in the installing folder

D:\





\* The installation folder of tomcat is called as <tomcat-home>

\* To start tomcat server use <tomcat-home>/bin/tomcat6.exe file

\* To launch home page of tomcat the procedure is

open browser window → type http://localhost:8080/ in address bar  
 ↑  
 Port no. of tomcat server

Procedure to change Port no. of tomcat server after installation:

Go to <tomcat-home>/conf/server.xml file and modify port attribute value of first <connector> tag → restart the server.

\* The browser window keeps every webpage in the buffer before displaying it for end user.

\* A buffer is a temporary memory which can hold the data for temporary period.

\* While giving new port no. to any software service use 1025 to 65535 range number because 1 to 1024 range numbers are busy for with OS services.

\* Programmer can supply Java Web application to tomcat server either in the form of directory or war file.

\* Programmer uses <tomcat-home>/webapps folder to deploy web applications.

- \* A specification is a document or reference that contains set of rules and guidelines to develop the software.
- \* There are two types of specifications.
  - (1) open specification (Any one can develop softwares based on this specification)  
Ex : JDBC specification, servlet specification, JSP specification and etc.
  - (2) proprietary specification (only specific company which has given specification is allowed to develop the software)  
Ex : .net specification.
- \* servlet, JSP, EJB, JMS, Java mail and etc are the sub specifications of JEE specification.
- \* servlet specification contains rules and guidelines to develop servlet container.
- \* JSP specification contains rules and guidelines to develop JSP container.
- \* EJB specification contains rules and guidelines to develop EJB container by using various sub specifications of JEE the web server and application server softwares will be developed and these server softwares contains various containers like servlet container, JSP container and etc...
- \* JEE and its concepts like servlets, JSP, EJB and etc are not installable softwares because they are just given as specification.
- \* Working with web server and its containers or application server and its containers is nothing but indirectly working with JEE and its concepts. (like servlet, JSP and etc..)
- \* Every software specification contains an API representing rules and guidelines. These rules are nothing but methods declared in the interfaces and abstract methods declared in the abstract classes. These guidelines are nothing but concrete methods defined in abstract classes and concrete classes.
- \* While developing software based on software specification rules will be implemented (must) and guidelines will be utilized (optional).

\* Servlet specification supplied API is the classes and interfaces of javax.servlet package and javax.servlet.http package.

\* The methods of interfaces and the <sup>abstract</sup> methods of abstract classes available in above two packages represents rules of servlet specification whereas the concrete methods defined in classes and abstract classes of above two packages represent guidelines of servlet specification

What is Servlets?

\* Servlets is an open/open source API specification that contains set of rules and guidelines to develop servlet container software.

\* Based on this servlet specification given by Sun micro systems different vendors develops different servlet container softwares and supplies them to programmers along with web server and application server software installation.

\* In earlier days servlet container used to call as servlet engine.

\* The tomcat serve the main name of servlet container is catalina.

\* Since JEE is not a installable software the tomcat server supplies sun micro system's servlet API in the form of <tomcat-home>/lib/servlet-api.jar and it gives <sup>its</sup> the servlet container software in the form of <tomcat-home>\lib\catalina.jar.

\* Every API software specification supplied API will be used in two angles.

(1) Vendor companies use specification API to develop the softwares.

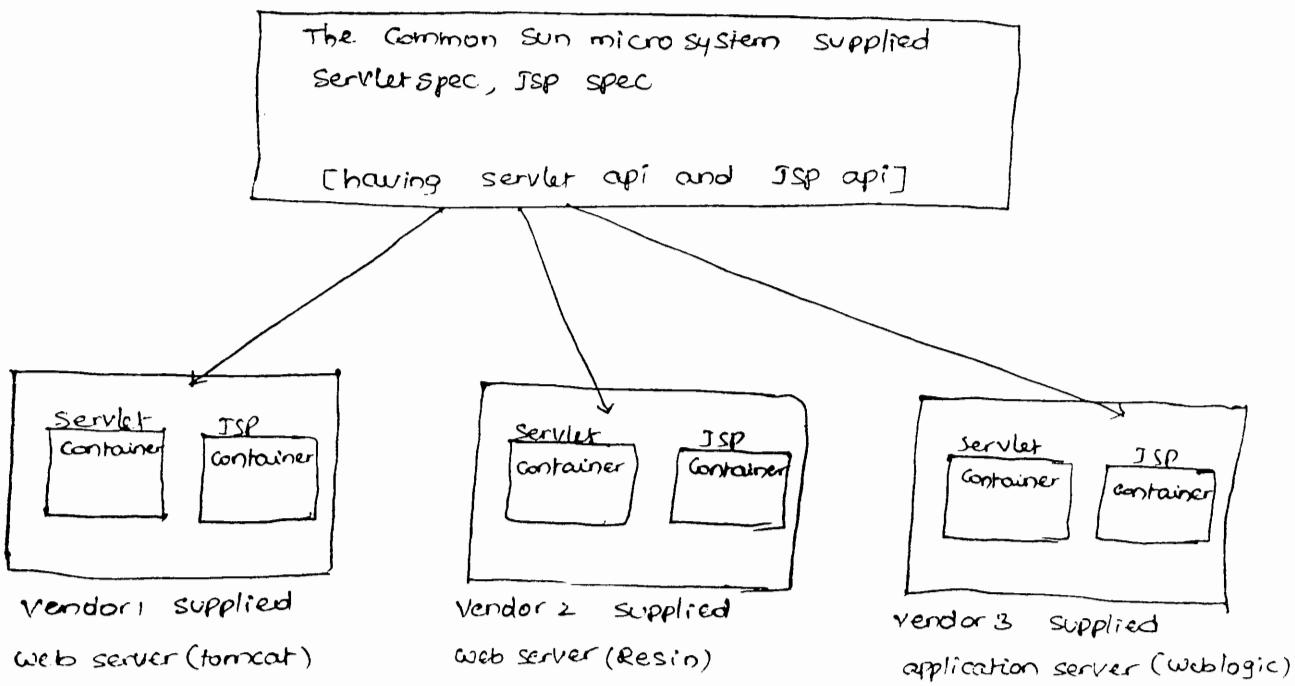
(2) Programmers use specification API to develop programs or applications.

\* The servlet API of servlet specification will be used by vendor companies to develop servlet container software whereas programmers use the same servlet API to develop servlet programs as web resource programs of web application. Due to this servlet container is always capable of executing servlet programs.

What is JSP?

\* It is sun micro system supplied an open specification having set of rules and guidelines to develop JSP container.

- \* JSP specification is enhancement of servlet specification so JSP container is also the enhancement of servlet container.
- \* In tomcat server JSP container is called as Jasper.
- \* The tomcat server supplies sun microsystems JSP api in the form of <tomcat-home>\lib\jsp-api.jar and supplies its own JSP container in the form of jasper.jar.
- \* The sun microsystems JSP api contains following packages
  - javax.servlet.jsp
  - javax.servlet.jsp.el
  - javax.servlet.jsp.tagext



\* Since all web server softwares and their container softwares given by different vendors are developed based on common Sun micro system supplied servlet specification, JSP specification so the way we work with all these web server softwares and their container softwares in Java based web applications development is going to be much similar.

## Server side Technologies

(required to develop server side web resource prgs of web apps)

Process Based

CGI - perl

Thread Based

servlets

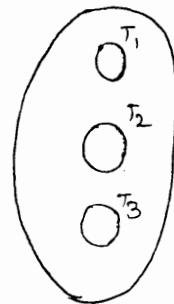
JSP

ASP

ASP.NET

PHP

OS managed process



$T_1, T_2, T_3$  are threads

\* A thread is a light weight process in OS managed process.

\* Every Java application contains two default threads.

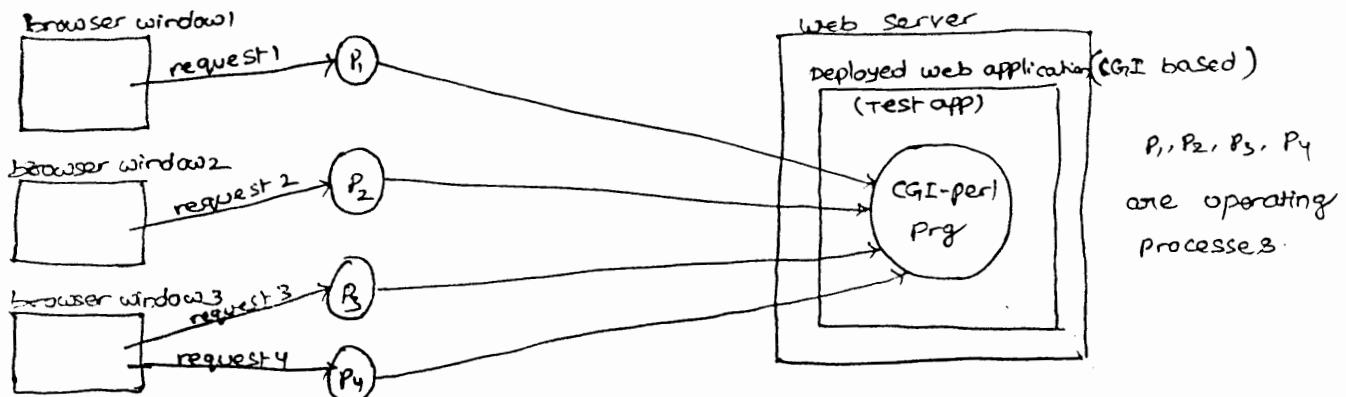
(1) main thread

(2) Garbage collector thread

\* CGI is a specification to develop server side web resource program  
so perl programming language will be used to develop the webresource programs based on CGI specification.

\* perl  $\rightarrow$  practical extraction reporting language

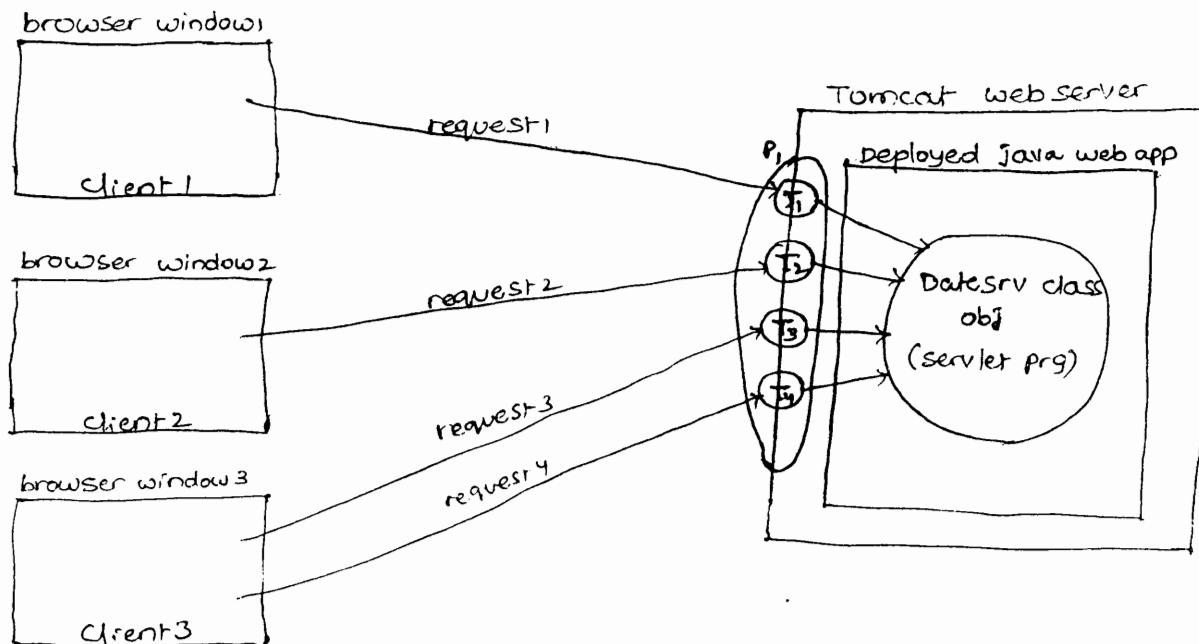
### Understanding Process based Serverside technologies. (CGI)



- \* The procedure of transferring control from one process to another process or from one thread to another thread is called as the scheduling based control jumping or context switching.
- \* Since OS processes are heavy weight processes so they take lot of time for control jumping or context switching.
- \* Due to this when more requests are given CGI based web application more OS processes will be created on one per request basis and performance of website will be degraded. (That means web application becomes non scalable web application).
- \* If application is giving same performance irrespective of increase or decrease in number of clients / no. of requests is called as scalable application.

\* Every server program is a Java class using servlet API.

Understanding thread based server side technologies like servlet



P1 → OS managed process representing the webserver startup

T1, T2, T3, T4 are threads started on servlet program related object representing request given by clients.

- \* The control jumping between threads of a process takes very much less time when compare to control jumping between two separate OS processes. So we can say thread based server side technologies give better

○ performance when compared to process based server side technologies.  
○ \* The web application that is developed based on thread based server side technologies is scalable application.  
○ \* servlet program is a single instance multiple threads based Java component of java web application as shown above. That means if 100 requests are given to single servlet program then the servlet Container creates only one object for that servlet program class but starts 100 threads on that object representing 100 requests as shown above.

### Definitions of servlet

- (1) Servlet is server side technology (basically specification) that allows the programmers to develop dynamic web resource programs or server side web resource programs in java based web application.
- (2) Servlet is a software specification given by Sun microsystems that provides set of rules and guidelines for vendor companies to develop the softwares called servlet containers.
- (3) servlet is a single instance multiple threads component based java component in java web application to generate dynamic web pages.
- (4) servlet is a server side technology (basically specification) that can enhance functionalities of web server software or application server software.
- \* servlet program is server independent because one servlet program can be executed in any java based web server or application server software without modifications.
- \* securing web application is nothing but applying authentication and authorization on the web application.
- \* checking the identity of a user is called as authentication.
- \* checking the access permissions of the user on particular resources of the project is called as authorization.
- \* for basic information on servlets, servlet container refer the page no.s 36 to 43.

## JEE 5 specification

Servlet 2.5 spec

JSP 2.0 spec

EJB 3.0 spec

:

## JEE 6 specification

Servlet 3.0 spec

JSP 2.1 spec

EJB 3.1 spec

:

\* In JEE module of Java all are specifications.

\* In JSE module of Java AWT, Swing are technologies, JDBC, JNDI are specifications.

\* The latest version of the Servlet API specification is 3.0 but industry is using Servlet 2.5 API specification.

\* Tomcat 6.0 supplies servlet container software based on Servlet 2.5 specification whereas Tomcat 7 use servlet container software based on Servlet 3.0 specification.

## The 3 important resources of Servlet API

javax.servlet.Servlet (I)

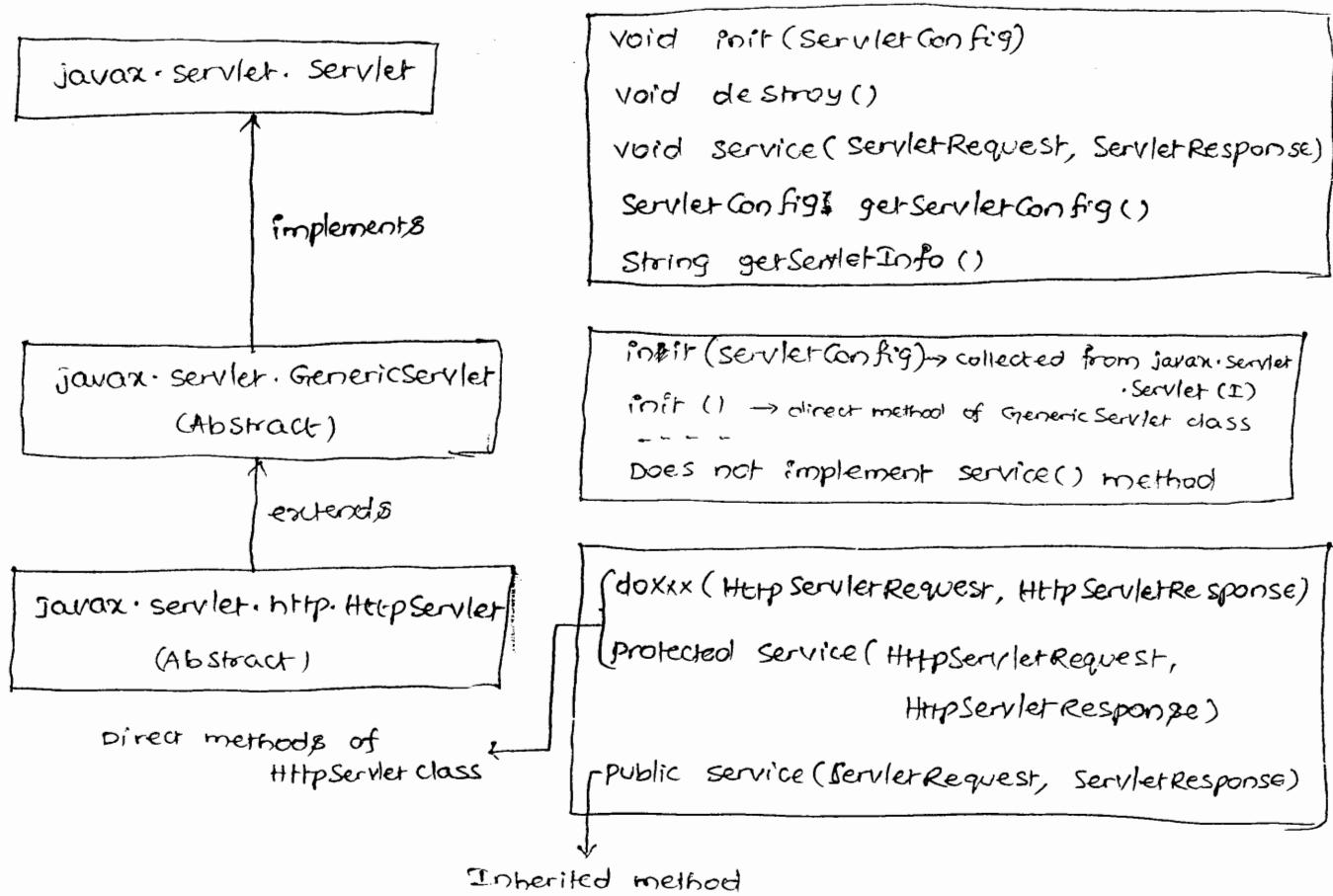
javax.servlet.GenericServlet (Abstract class)

javax.servlet.http.HttpServlet (Abstract class)

\* In Java an abstract class can contain only abstract methods or only concrete methods or mix of both.

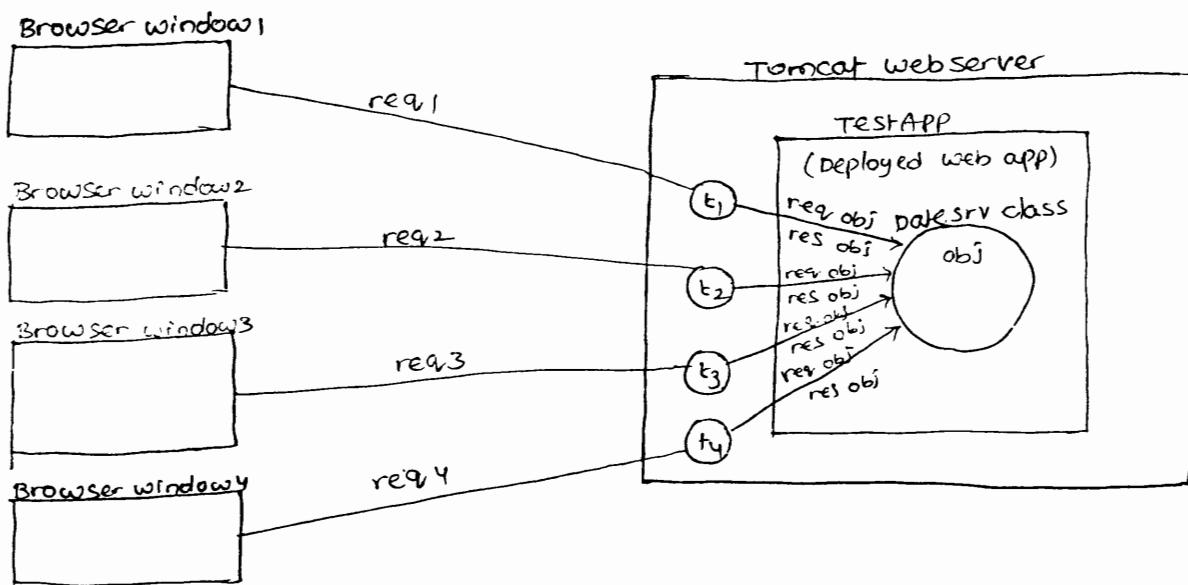
\* The above give predefined HttpServlet class is an abstract class containing no abstract methods.

\* If you want to make methods/ logics of one Java class accessible only through subclasses then make that class as abstract class even though that class contains only concrete methods.



- \* Every servlet program is a java class that is developed based on servlet api. There are three ways to develop servlet program.
- (1) Take a java class implementing `javax.servlet.Servlet` Interface and provide implementation for all the 5 methods of that interface.
- (2) Provide Take java class extending from `javax.servlet.GenericServlet` class and provide implementation for `service()` method.
- (3) Take java class extending from `javax.servlet.HttpServlet` class and override one of the 7 `doXXX()` methods or one of the two `service(-,-)` methods.
- \* In the above said three approaches the overridden/implemented `service(-,-)` / `doXXX(-)` programmer places request processing logic that generates web page by processing the request. But programmer never calls these methods manually but servlet container calls these methods automatically for every request given by client to servlet program.

- \* Programmer just supplies his servlet program related java class to servlet container then onwards servlet container is responsible to manage the whole life cycle of servlet program. (from object birth to death every operation will be taken care by container).
- \* For all the request given to servlet program the servlet container creates one object but for every request given to Servlet program, the Servlet Container creates one separate request, response objects and calls service(-,-) method by keeping these requests, response objects as argument values.



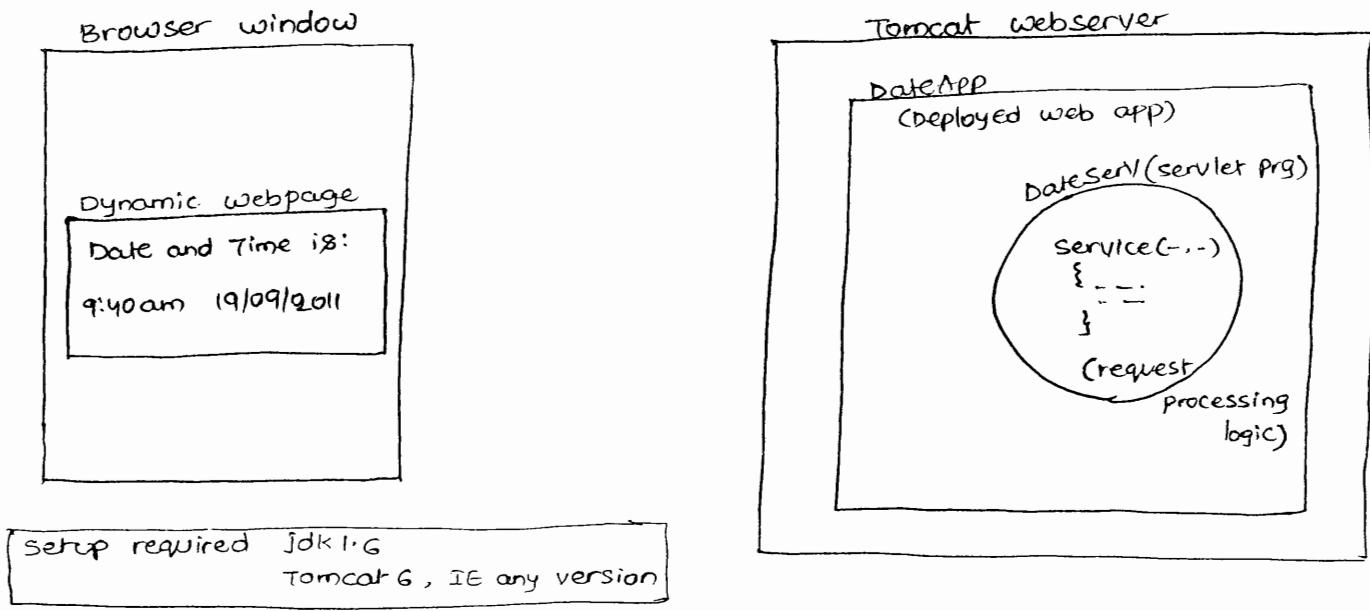
- \* servlet program uses request object to read details from the request like form data.
- \* servlet program uses response object to send response content to browser window through web server.

When 10 requests are given to a servlet program from single or different browser windows (clients)

- servlet container creates only one object for servlet program related java class.
- servlet container creates 10 threads on servlet program object representing 10 requests.
- servlet container creates 10 sets of request, response objects and calls service(-,-) or methods for 10 times having request, response objects as argument values.

\* To make our servlet program java class visible to servlet container the java class must be taken as public class.

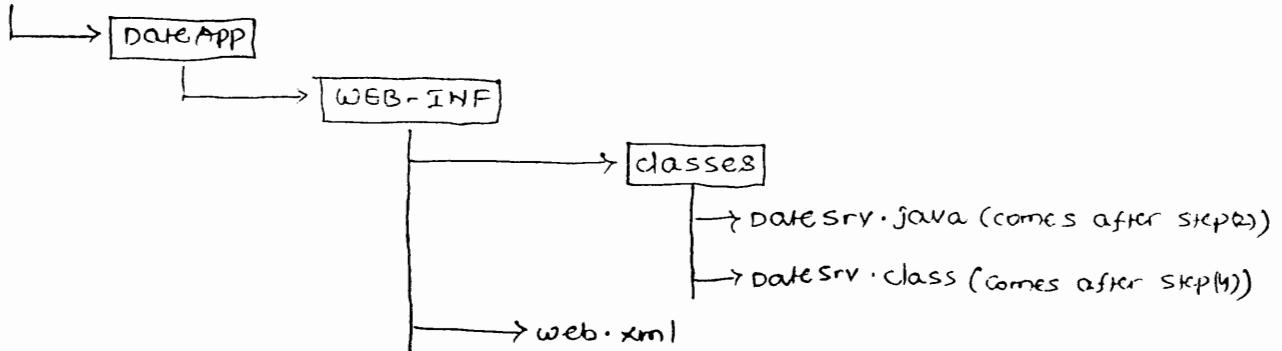
Procedure to develop first java web application by taken servlet program as webresource program



Step(1): create the following deployment directory structure or packing directory structure to combine multiple webresource programs into single unit.

NOTE: A web application is a directory or war file containing multiple web resource programs. (one or more).

E: |



\* Here DateApp folder is a webroot folder.

\* After deployment this DateApp becomes name of the web application.

\* The web application packing or deployment directory structure is common for all the servers and can be created in any location of Computer.

Step(2): develop servlet program (datesrv.java) save that one in WEB-INF  
classes folder of web application.

```
//Datesrv.java
import javax.servlet.*;
import java.io.*;
public class datesrv extends GenericServlet
{
    // implement service (-,-)
    public void service (ServletRequest req, ServletResponse res) throws
        ServletException, IOException
    {
        // Set response Content type
        res.setContentType("text/html");
        // get stream obj
        PrintWriter pw = res.getWriter();
        // write req processing logic
        java.util.Date d1 = new java.util.Date();
        // write response content to browser window through webserver
        pw.println ("Date and time is: " + d1.toString());
        // close stream object
        pw.close();
    } // service (-,-)
}
```

/\* class

req(1): Represents the servlet Container supplied request object

res(2): Represents the servlet Container supplied response object.

text/html: MIME (Multipurpose Internet mail Extension/Content type).

res.setContentType("text/html");

\* Represents that passes instructions to browser window through web-server that this servlet program generate html code base response.

## other mime types

\* text/html

\* application/ms word

\* application/ and etc

printwriter pw = res.getWriter();

\* Here pw is a stream object pointing to response object getWriter() called on res object returns the printwriter stream object.

NOTE: A stream object can represent either file or object as destination

pw.println(" --- ");

\* This method makes the stream object pw writing data to the destination object response(res). This response object passes that data (argument value of println("...") method) to webserver and this webserver writes data to browser window as http response in the form of a web page.

pw.close();

\* closes the stream connection with response object.

\* Add the tomcat server supplied servlet-api.jar (contains servlet api) to classpath.

My Computer → properties → advanced variables → environment variables →

variable name: CLASSPATH

value : D:\Tomcat 6.0\lib\servlet-api.jar; <other existing jar files>;

→ OK → OK → OK

NOTE: Since servlet-api is not part of jdk software and since our servlet program is using servlet api to make java compiler (javac) recognising api the above jar file is added to the classpath.

Step(3): Compile our servlet program source code.

E:\DateApp\WEB-INF\classes > javac DateSrv.java

Step (4): Develop web.xml file having the servlet programs configuration.

NOTE :

- (1) A program or file of application is called as resource of the application.
- (2) servlet program is called as the web resource program of the web application. All servlet programs must be configured in web.xml file.
- (3) specifying the details of certain resource program and making underlying software like Container, Server and etc recognizing the resource programs is called as resource configuration. By configuring servlet programs in web.xml file we make servlet Container recognising servlet programs.

web.xml

<web-app>

    <Servlet>                                                  → logical name/ object name for our servlet class when servlet container creates the object.  
        <Servlet-name> abc </Servlet-name>

        <Servlet-class> DateSrv </Servlet-class>

    </Servlet>                                                  → java class that is acting as servlet program

    <Servlet-mapping>

        <Servlet-name> abc </Servlet-name>                                  → must match above with above logical name

        <Servlet url-pattern> /test </url-pattern>

    </Servlet-mapping>                                          → url pattern of servlet program

</web-app>

\* servlet program will be identified in the web.xml file through its logical name (abc). more ever servlet container uses this logical name as the object name when it creates object for our servlet program java class.

\* The webserver servlet container and the web resource programs of web application and clients identifies the servlet program through its url pattern (/test).

\* web.xml file is called as deployment descriptor (dd) file because servlet container reads and verifies the entries of web.xml file the moment web application is deployed.

\* web.xml file is called as Configuration file because it contains servlet program configurations.

\* physical presence of servlet program in WEB-INF/classes folder is not sufficient to make servlet container to recognize servlet program.  
It must be configured in web.xml file.  
NOTE: step(1) to step(4) represents web application development.  
Step(5): Start the tomcat server.

Tomcat-home\bin\tomcats

Step(6): Deploy the above DateApp web application

copy e:\DateApp folder to Tomcat-home\webapps folder.

NOTE: Step(5) and Step(6) performs the deployment of web application.

Step(7): Test the web application  
open browser window → type  $\text{http://localhost:2020/DateApp/test}$   
Protocol ↑ host name & port no. of tomcat server  
web application name/Context ← ↓  
path / context route URL pattern of DateSrv servlet program

NOTE:

\* URL pattern given to servlet is useful, to hide the servlet related program class name from end users.

\* Java web applications are WODA (write once deploy Anywhere) applications because

(1) The deployment directory structure is common for all web servers.

(2) The web resource programs like servlet, JSPs are server independent programs.

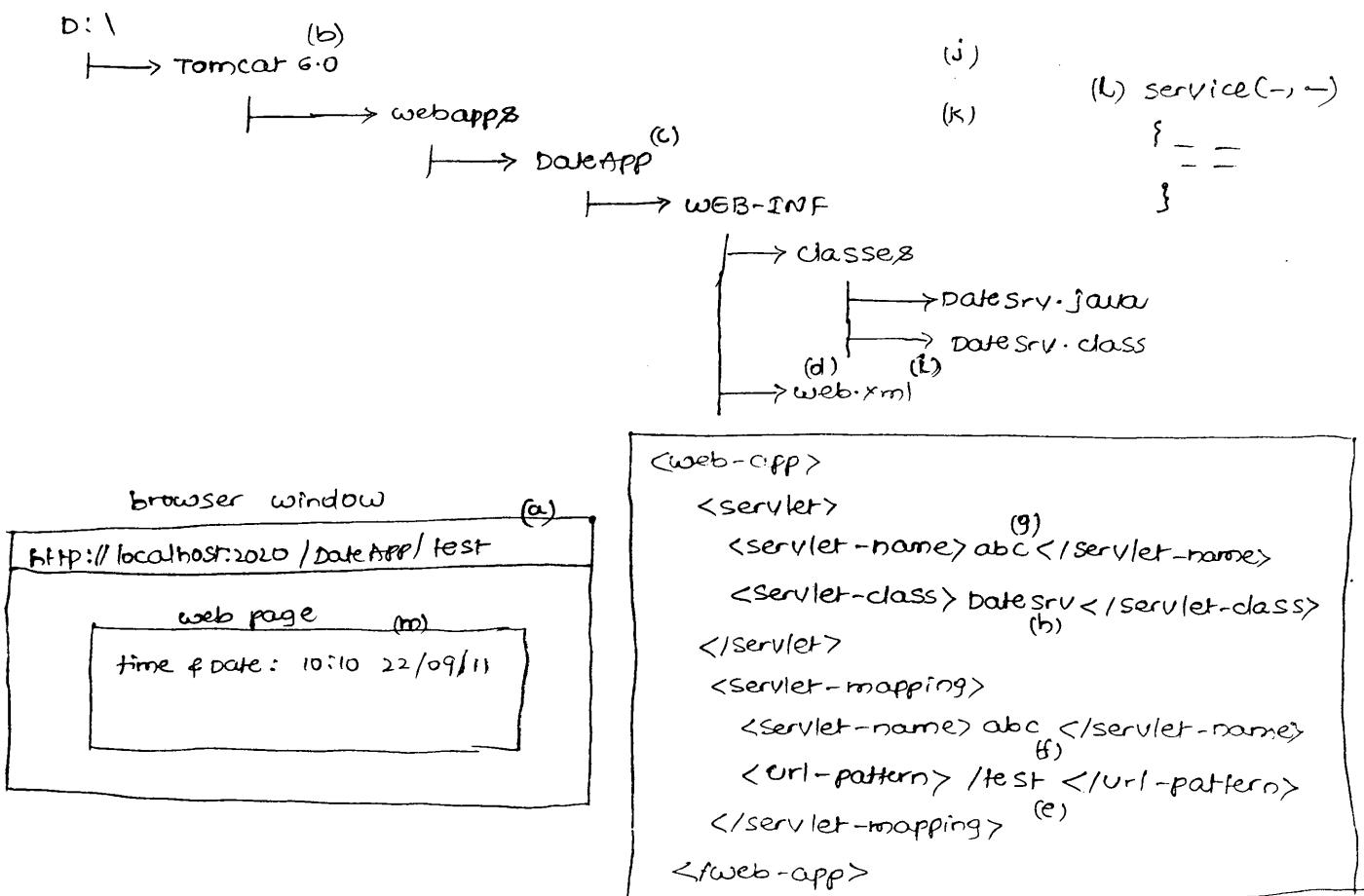
Can you explain flow of execution from request to response generation with the aid of an example application?

with respect to the given code

(a) End user types request URL in the address bar of browser window to generate request to servlet program.

(b) Based on localhost:2020 content of request url the request goes to the tomcat webserver of local machine.

(c) Based on DateApp word of request url the request goes to the



deployed DateApp web application of web server.

- (d) The servlet Container uses `web.xml` file entries to locate the servlet program that is requested.
- (e), (f), (g), (h) Based on the url pattern `/test` the logical name abc the servlet Container gathers the class name of servlet program (`DateSrv`).
- (i), (j) servlet Container loads `DateSrv` class from `WEB-INF\classes` folder of web application.
- (k) servlet Container creates and locates `DateSrv` class object. If created the servlet Container finishes initialization process on our servlet class object.
- (l) servlet Container creates one set of request, response object for the current request and calls `service (-, -)` method by keeping them as argument values.
- (m) The `service (-, -)` method processes the request and generates the output.

(ii) This output goes to browser window through webserver as http response in the form of dynamic web page.

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

\* In the above statement we are getting access to PrintWriter stream object that is available as built-in Stream object of res object. This Stream object points to res object as OutputStream object so we can use this Stream object to write some output content to response object from servlet program.

\* JavaIostream objects can perform read and write operations on the files/objects.

\* The System.out.print statements generated output from servlets will appear on server console whereas the pw.println statements generated output will come on webpage of browser window.

\* modifications done in web.xml file of deployed web application will be recognized by server automatically whereas the modifications done in servlet program source code will be reflected only after recompilation of servlet program and reloading of the web application.

\* To reload web application in Tomcat web server the procedure is

open Tomcat home page (<http://localhost:2020/> → Tomcat manager →

user name : admin  
password : admin } chosen during installation

→ DateAPP application → Reload.

\* Servlet container uses 0-arg constructor to create object of our servlet class so make sure that our servlet class is having 0-arg constructor directly or indirectly. Directly means programmer should place 0-arg constructor explicitly. Indirectly means servlet program deals with the javac generated default 0-arg constructor.

\* The service method placed in servlet program will be called by servlet container for every request that is given to servlet program from

any browser window (client).

javax.servlet.ServletRequest (I)

↑ extends

javax.servlet.http.HttpServletRequest (I)

javax.servlet.ServletResponse (I)

↑ extends

javax.servlet.http.HttpServletResponse (I)

\* The req object of service(--) method is not the object of javax.servlet.ServletRequest interface, it is the object of servlet Container supplied Java class implementing javax.servlet.ServletRequest interface directly or indirectly. But programmer never specifies this class name in servlet program because its ~~class~~ name changes based on the webserver / servlet container we use. In tomcat server req object class name is "org.apache.catalina.connector.RequestFacade".

\* The res object of service(--) method is not the object of javax.servlet.ServletResponse interface, it is the object of servlet Container supplied Java class implementing javax.servlet.ServletResponse interface directly or indirectly. In Tomcat server the res object class name is "org.apache.catalina.connector.ResponseFacade".

\* To know req, res objects class names in any server call getClass() methods on those objects from service(--) method of servlet program as shown below.

```
pw.println("<br>req obj class name is:" + req.getClass());  
pw.println("<br>res obj class name is:" + res.getClass());
```

\* All the request coming to our servlet program will use only one object of our servlet class but every request gets separate request, response objects and also starts one separate thread on servlet class object (refer).

\* To prove this practically write following code in the service(--) method of our servlet class.

```

pw.println("<br> req obj class name is :" + req.getClass());
pw.println("<br> res obj class name is :" + res.getClass());

gives separate values for each request
    pw.println("<br><br> hash code of req obj : " + req.hashCode());
    pw.println("<br><br> hash code of res obj : " + res.hashCode());

gives separate for each request
    pw.println("<br><br> current req thread name is :" + Thread.currentThread()
               .getName());
    pw.println("<br><br> hash code of our servlet class obj : " + this.hashCode());
}

try {
    Thread.sleep(40000);
}
catch (Exception e) {
    e.printStackTrace();
}

```

NOTE: once server is down all objects (including our servlet class objects) will be destroyed.

### How many types of Servlets are there?

- \* There is a possibility of developing n-types of servlets like http servlet, FTP servlet, SMTP servlet and etc..
- \* Since the entire internet and all webserver softwares are given based on the protocol http the programmers prefer developing their servlet programs as http servlet programs.
- \* javax.servlet.GenericServlet is not a separate type of servlet. It is given as common super class for all protocol specific servlet classes like http servlet and etc..
- \* As of now servlet api is giving only one sub class for GenericServlet class that is HttpServlet class because all servers are there supporting only http protocol.

### How a servlet program developed based on GenericServlet class can work with HttpServletRequest and HttpServletResponse ?

- \* Since GenericServlet class is common super class for protocol specific servlet classes. It can take HttpServletRequest but it can not gather all the

details from that `HttpServletRequest`. Whereas the servlet program that is developed based on `HttpServlet` class can gather all the details of `HttpServletRequest`.

- \* Even though there are lot of details in servlet program but the servlet program will be identified with its URL pattern.
- \* we can provide multiple URL patterns for a single servlet program in `web.xml` file as shown below.

```
<web-app>
  <servlet>
    <servlet-name> abc </servlet-name>
    <servlet-class> datesrv </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name> abc </servlet-name>
    <url-pattern>/test </url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name> abc </servlet-name>
    <url-pattern>/test1 </url-pattern>
  </servlet-mapping>
</web-app>
```

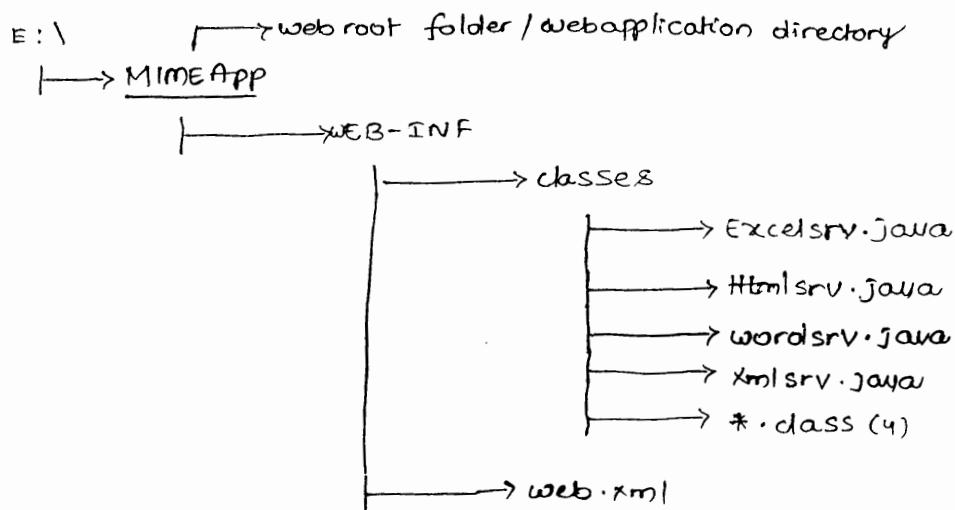
- \* If web application is having multiple servlet programs then all the servlet programs must be configured in `web.xml` file having URL patterns.
- \* By specifying different MIME types as response content types we can make servlet programs sending web pages to browser window in different formats.
  - Every software installed in your computer will provide MIME type or content type for its files, to know them we can use the windows registry editor (regedit) tool
  - Browser window can display web pages in different formats based on the response content type of the output that is generated by web resource programs. To know MIME type or content type of any windows operating system managed file, the process is

Start → Run → regedit → HKCU\CLASSES\ROOT → choose file extension  
→ get Content type from right side panel.

.doc (ms word) ----- application/msword  
 .xls (msexcel) ----- application/vnd.ms-excel  
 .html ----- text/html  
 .xml ----- text/xml  
 .bmp ----- image/bmp  
 .avi ----- video/avi  
 and etc...

Develop a java web application having multiple servlet programs and each servlet should generate a different format based web page.  
 (servlet programs or Htmlsru, Excdsr, Wordsrv, Xmlsrv)

Step(1): prepare the deployment directory structure of web application.



Step(2): Develop the source code of above servlet programs.

\* For the source code of Htmlsru.java, Wordsrv.java, Xmlsrv.java, Excdsr.java refer page nos 58 and 59 of the booklet.

NOTE: The approach no. (1), (2) based servlet programs work with ServletRequest, ServletResponse objects where as the approach no. (3) based servlet program works with HttpServletRequest object and HttpServletResponse object.

NOTE: The simple ServletRequest, ServletResponse objects can't work with all the facilities of the protocol Http where as HttpServletRequest, HttpServletResponse objects can work with all the facilities of protocol HTTP.

Step(3): Add servlet-api.jar file to classpath.

Step(4): Compile the source files of all the servlets programs.

E:\MIMEApp\WEB-INF\classes > javac \*.java

Step(5): Configure all the four servlet programs in web.xml file having four different URL patterns.

Refer the web.xml file of page nos 59 and 60

Step(6): Start the server (Tomcat)

Step(7): Deploy the web application.

copy E:\MIMEApp folder to Tomcat-home\webapps folder

Step(8): Test the web application.

open browser window → type these URLs

http://localhost:2020/mIMEApp/hturl

http://localhost:2020/mIMEApp/ldurl

http://localhost:2020/mIMEApp/xlsurl

http://localhost:2020/mIMEApp/xmlurl

\* A servlet program can generate both static and dynamic web pages. In the above application all the servlet programs are designed to generate static web pages.

\* The WEB-INF folder of Java webapplication deployment directory structure is called as private directory because it is not visible outside the web-server and it is visible only to that web server where our java web-application is deployed. more ever the web resource programs that are placed in WEB-INF folder or in its sub folders (like classes folder) will be recognized by servlet container only when they are configured in web.xml file.

\* Since we place our servlet programs in WEB-INF\classes folder so they must be configured in web.xml file.

\* If multiple servlet programs of a web application are configured with ... pattern the last servlet in that list (web.xml order) will be executed for that url pattern based request generation.

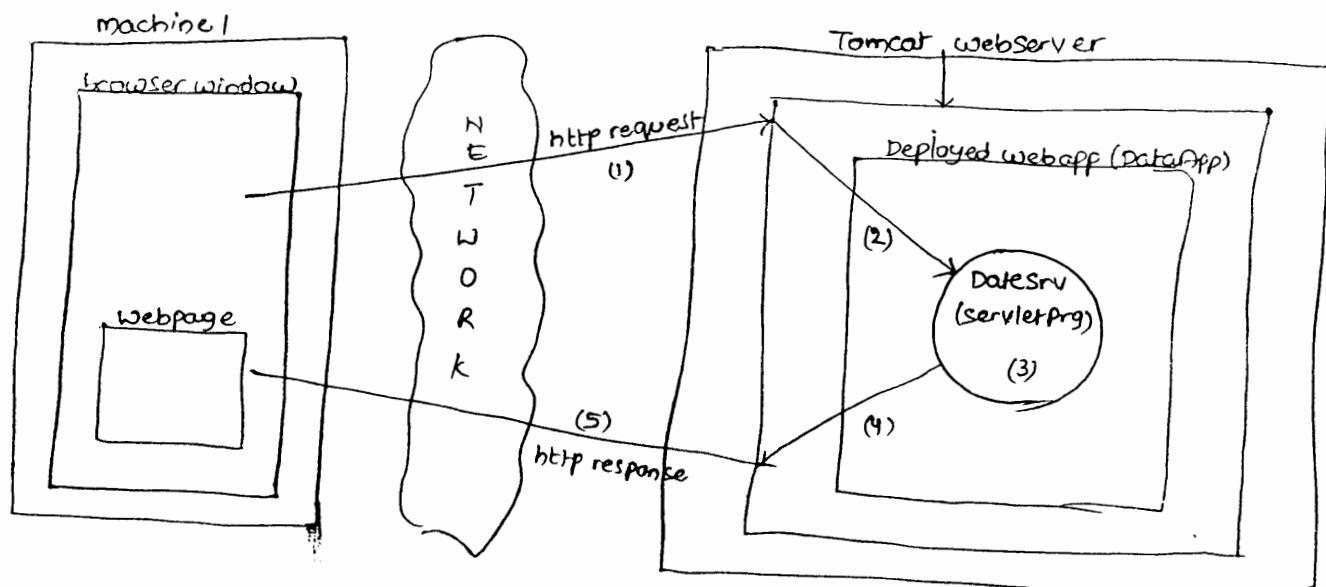
## understanding HTTP

- \* Protocol defines set of rules that are required to get communication between two parties.
- \* HTTP is a application level protocol that runs over network level protocol called TCP/IP having set of rules to get communication between browser window and web server.
- \* Application level protocols defines a set of rules that are required to get communication between two softwares or software applications.

Ex: HTTP, JDBC:odbc, Jdbc:oracle, SMTP and etc...

- \* Network level protocol defines set of rules to get communication between two physical computers of network.

Ex: TCP/IP



Request url:

http:// machine 10 : 2020 / DateApp / test ? sno = 101 & sname = raja

(query string having req params)

- \* The query string appended to the request URL sends data to destination web resource program (like servlet) along with the request.
- \* When request URL is typed in the address bar of browser window the browser window generates one http request having lots of details and we can remember these details as **H<sup>r</sup>p details**.

H ----> http request method (GET/POST/DELETE/.....)

H -----> http request headers (like user-agent, accept, accept-language....)

P -----> request path (http://machine10:2020/Date-App/test)

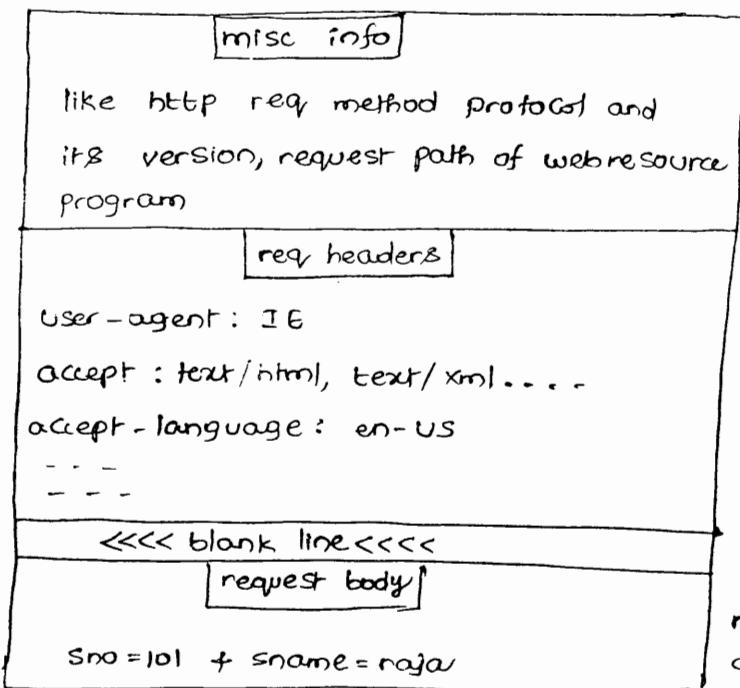
P -----> request parameters / query string (sno=101 & sname=naraja)

↓                            ↓  
request parameter      request parameter value  
name

\* Browser window is responsible to generate http request having multiple details based on the request URL that is generated.

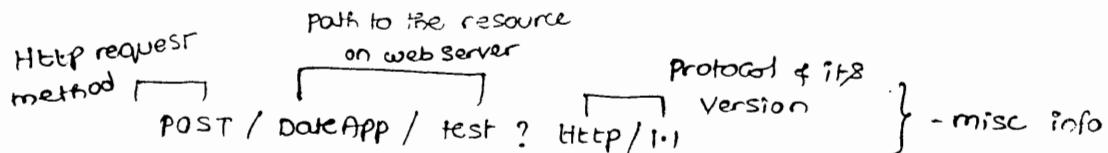
### The Structure of http request

#### http request



request body is also called as request payload

### An example http request with content



Host: www.natarajsoft.com

user-agent: mozilla/ IE 5.0

accept: text/html, text/xml, image/jpeg, application/msword, \*/\*

accept-encoding: gzip, deflate

Accept-charset : ISO-6431  
 Keep-Alive : 300  
 Connection : Keep-Alive  
 <<< blank line <<<<  
 sno=101 & sname=raja } request body / payload having  
 request parameters.

- \* "GET" method based request can send limited amount of data along with request (max of 256 kilo bytes).
- \* "POST" method based request can send unlimited amount of data to server along with the request.

What is the difference between request parameters and request headers of http request?

- \* Both are there to send input values to target web resource program like servlet program along with the request. The differences are:

#### Request parameters

- \* Represents end user supplied input values to web resource program.
- \* End user supplies these values either through form page or by appending query string in the URL.
- \* Request parameter names and values are not fixed.
- \* Multiple request parameters can have same names.
- Ex: sno=101, sname=raja and etc.

#### Request headers

- \* Represents the browser window supplied input values to web resource program.
- \* Browser window internally adds request headers and their values to http request automatically.
- \* Request header names are fixed, predefined and their values will be generated by browser window through browser settings.
- \* Request header names are unique names.
- Ex: user-agent : IE  
accept-language : en-us

\* For related information about all http request headers refer page no. 48.

- \* To know various details about http request page nos 46 to 49
- \* servlet program can use all the details of http request as input values while processing the request.
- \* A Simple `servletRequest` object can gather only misc info, req param values from http request given by client (browser window).
- \* A `HttpServletRequest` object of servlet program can gather all the details from http request (including headers) given by client.

### Request URL

`http://localhost:2020/DateApp/test ? sno=101 & sname = roja & sadd = hyd & sadd = vizag`  
req params

Different approaches of gathering req parameters values/ request body values from

#### Http request being from servlet program

##### Approach(1): (by using `req.getParameter(-)`)

Code in `service(..)` of servlet program

`String s1 = req.getParameter("sno");` → gives "101"

`String s2 = req.getParameter("sname");` → gives "roja"

`String s3 = req.getParameter("sadd");` → gives "hyd"

NOTE: In this approach we must know req parameter name to get its value, if request parameter is having multiple values than it gives only first value.

##### Approach(2) (by using `req.getParameterNames()`)

Code in `service(..)` of servlet prg

```
Enumeration e = req.getParameterNames(); // given enumeration obj pointing to
                                         // list data structure having req
                                         // param names
```

`while (e.hasMoreElements())`

{

`String pname = (String)e.nextElement();` // gives each req param name

`String pvalue = req.getParameter(pname);` // gives each req param value

`PW.println(pname + " → " + pvalue);`

}//while

Output is  
`sno => 101`

`sname => roja`

`sadd => hyd`

\* In this approach we can get req param values without knowing their names.

\* If one req param is having multiple values this gives only first value.

### Approach(3) (By using req.getParameterValues(-))

```
String s[] = req.getParameter("sadd");
for(int i=0; i<s.length; i++) { pw.println(s[i] + "..."); } //gives hyd...vizag values
```

```
String s1 = req.getParameterValues("sadd")[0]; //gives hyd
String s2 = req.getParameterValues("sadd")[1]; //gives vizag
```

\* This approach is useful to gather multiple values of single req parameter.

\* To read a req param values from http request we can use either servletRequest obj or HttpServletRequest obj.

\* While opening tomcat manager through browser window if user name and password related problems are raised, then add following two lines of code in Tomcat-home/user.xml

```
<role rolename="manager"/>
<user username="admin" password="admin" roles="managers"/>
```

\* res.setContentType("text/html");

\* According to the above statement setContentType is the method that is declared in javax.servlet.ServletResponse Interface and that method is implemented in container supplied java class implements this ServletResponse interface.

\* In case of Tomcat the implementation class name is responseFacade.

\* String s1 = req.getParameter("sno"); → gives 101

\* In the above statement getParameter is the method that is declared in predefined servletRequest interface and implemented in servlet Container supplied java class that implements servletRequest interface (In case of tomcat this class name is RequestFacade).

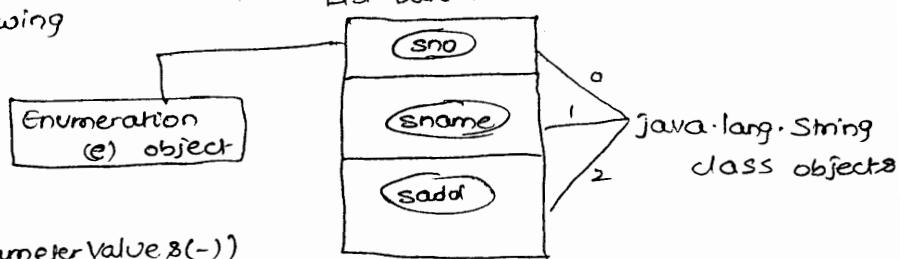
Ex: interface xyz

```
    public void x();
}
class Test implements xyz
{
    public void x()
    {
    }
    public void y()
    {
    }
}
```

```
xyz obj1 = new Test(); // Here obj1 is reference variable of interface but not object.
obj1.x();           // obj1 is object of implementation class Test
```

\* When interface reference variable points to its implementation class object then it becomes object of implementation class and it is not the object of interface.

List Data Structure



\* So when you call method on this reference variable then interface method will not be executed but the method defined in implementation class will be executed.  
Different approaches of gathering request header values from Http request being from servlet program

#### Approach(1): (by using req.getHeader(-))

Code in service(-,-) of servlet Program

```
String s1 = req.getHeader("User-Agent"); //gives browser s/w name
```

```
String s2 = req.getHeader("Accept"); //gives the mime types supported by browser  
window like text/html, text/xml etc...
```

\* In this approach we must know request header name to get its header value.

#### Approach(2): (by using req.getHeaderNames())

Code in service(-,-) of servlet program

```
Enumeration e = req.getHeaderNames(); ①
```

```
while (e.hasMoreElements())
```

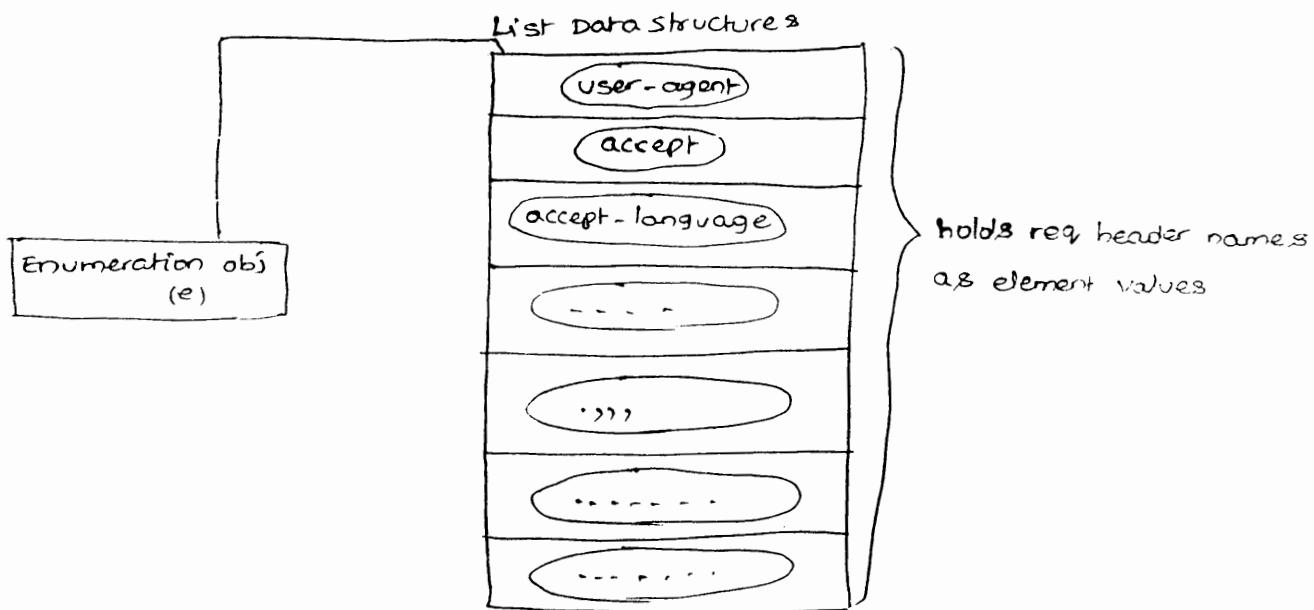
```
{
```

```
    String hname = (String) e.nextElement(); ②
```

```
    String hvalue = req.getHeader(hname); ③
```

```
    pw.println(hname + " => " + hvalue);
```

```
}
```



- (1) Gives enumeration object pointing to list data structure having request header names
- (2) gives each request header name
- (3) gives each request value.

- \* This code gives all request header names and values.
- \* The above code must be executed with the support of HttpServletRequest object.
- \* Through browser language setting we can set or modify accept-language request header value.
  - Internet Explorer → tools menu → Internet options → languages → add → choose language.
  - In Netscape Navigator Edit → preferences → Navigator → languages → add → choose language.

Q) How to gather current browser window software name being from Servlet program?

(A) use user-agent request header that holds browser software name as shown below.

#### In service (-) method

```
(1)     pw.println("<br><br> the current browser window"+req.getHeader(user-agent))  
(2)                                         ↓  
(3)                                         request header  
                                         name
```

Q \* The URL pattern of servlet programs is technically called as servlet path.

\* we can gather the misc info from `HttpRequest` by calling various `getxx` methods on `request` object as shown below.

```
① //example request url: http://localhost:2020/DateApp/test? sno=10  
② pw.println("<br> misc info of http request");
```

//methods available in javax.servlet.ServletRequest (T)

```
o      pw.println("<br>req_content_length:" + req.getContentLength());
```

// gives request data length in bytes like 345 bytes (if not known gives -1)

```
pw.println("<br>req Content type:" + req.getContentType());
```

//gives req Content type like text/html (if not known gives null)

```
pw.println("<br>req protocol: " + req.getProtocol());
```

//gives http/1.1

```
pw.println("<br>req. scheme:" + req. getScheme());
```

// gives HTTP

```
pw.println("<br>req. browser window machine IP address:");
```

```
+ req.getRemoteAddr());
```

Gives 127.0.0.1

```
pw.println("<br>browser window machine host name:" + req.getRemoteHost());
// gives current computer name other wise IP address
pw.println("<br>browser window port no:" + req.getRemotePort());
// gives 2922 for Nestcafe , 2926 for IE
pw.println("<br>server name" + req.getServerName());
// gives the domain name typed in the URL like local host
pw.println("<br>server port" + req.getServerPort());
// gives 2020

//methods available in javax.servlet.HttpServlet Request (I)

pw.println("<br>context path:" + req.getContextPath());
// gives DateApp
pw.println("<br>req method is:" + req.getMethod());
// GET
pw.println("<br>req Path info is:" + req.getPathInfo());
// gives additional information kept in req url (otherwise null)
pw.println("<br>query String is:" + req.getQueryString());
// gives sna=101
pw.println("<br>request uri :" + req.getRequestURI());
// gives /DateApp/test
pw.println("<br>request url:" + req.getRequestURL());
// gives http://localhost:2020/DateApp/test
pw.println("<br>server path :" + req.getServletPath());
// gives /test
```

\* servlet program can avoid browser dependent tags while generating html tags based response if by knowing browser software name through the request header user-agent.

+ when web resource program (like servlet program) sends response to browser window the generated Http response contain multiple details including response content. Those details can be remembered as seth details.

S → Response status code (100 - 599)

100 - 199 → Info

200 - 299 → Success

300 - 399 → Redirection

400 - 499 → Incomplete web resource

500 - 599 → Server Error

C → Content type (webpage Content)

H → response headers

like ContentType, ContentLength, refresh and etc.

\* Response status code indicates the status of generated response to display on the browser window.

\* Every generated Http response contains one http status code, default status code is 200.

\* If web resource program generates warnings based web page then the status code is 100-199.

\* If web resource program generates successful web page then the status code is 200-299.

\* If request given to one web site is forwarded to another website then the status code will be 300-399.

\* If our web resource program is incomplete or invalid to process the request then the status code will be 400-499.

\* If server fails to execute our web application the status code will be 500-599.

\* 400-599 are error status codes, using them the programmer can debug the problems related to web application execution and server.

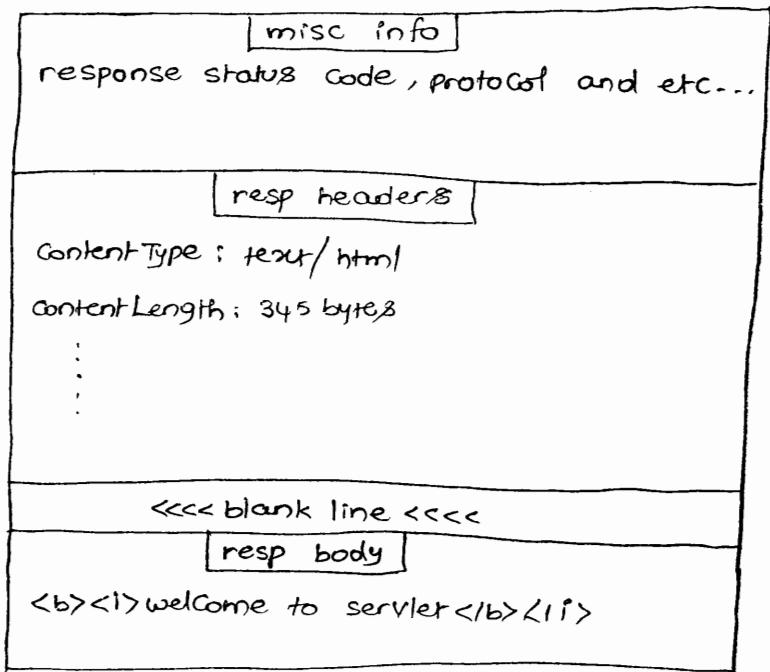
\* 100-399 indicates success response status codes that means they display web pages on browser window having output content so these status codes will not appear on the webpages.

\* For related information on http response status codes refer page no 49 and 50.

\* Response headers provide instructions to browser window through web server towards displaying web pages on the browser windows.

\* The Content that we have placed in pw.println statements becomes response body http response.

## Http Response Structure



resp body also called as resp content / resp payload.

### Http response with example content

Http / 1.1      200      ok      → Http status code      → misc info

Set -Cookie : JSESSIONID = 148937ABEA179EA

Content-type : text/html

Content-length : 1354

Date : Fri Feb 2008 8:40 34 GMT

Server : Apache - Tomcat /5.0

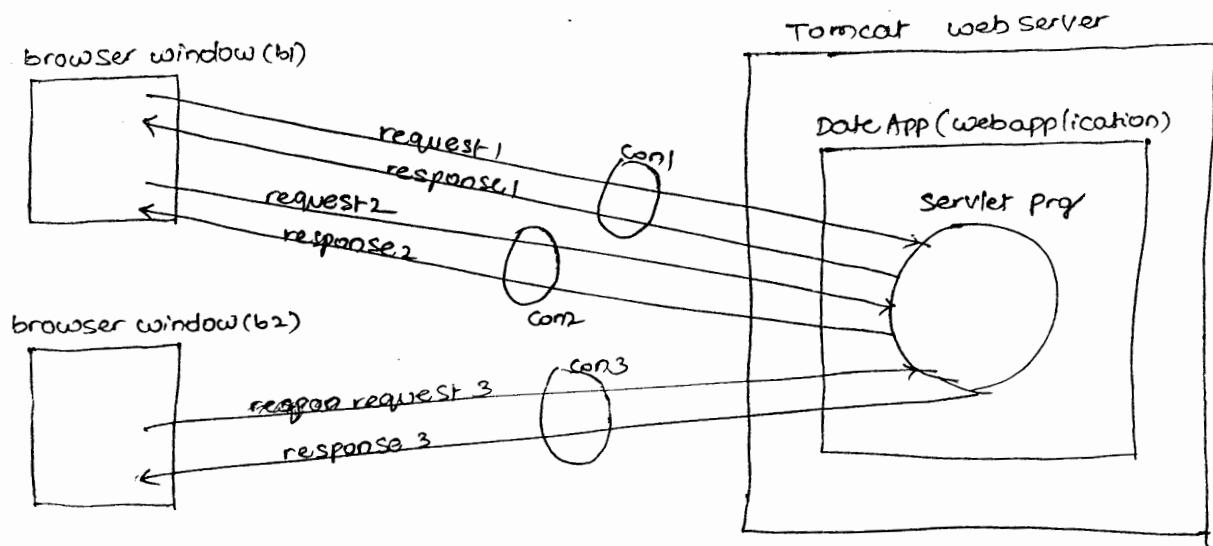
Connection : close

<<< blank line <<<

<html> <body> welcome servlet </body> </html> → response body

↓  
response content / response body

\* For every request generated by browser window one new connection will be created between browser window and web server and this connection will be closed automatically once request related response comes back to browser window.



\* For related information on response headers refer page no.s 50 and 51.

### The list of all http request headers

accept, accept-encoding, authentication, connection, cookie, host, if-modified-since, referrer, user-agent, keep-alive, accept-charset and etc..

### List of http response headers

location, refresh, set-cookie, cache-control / Pragma, content-encoding, content-length, Content-Type, last-modified, date, server, connection and etc...

### How to make browser window refreshes its web page automatically at regular intervals?

\* Give instructions to browser window from servlet through server by using response header called refresh as shown below.

```
res.setHeader ("refresh", "10");
                ↓
                time in seconds
```

By default every web resource program generated output/response content will be stored in the buffer before it is getting displayed on browser window as web page content. Due to this browser window may show old output collected from the buffer even though the web resource program of web application is capable of generating new and updated output.

\* To solve the above problem instruct browser window for not storing the output in the buffer from web resource program through web server, by using response headers as shown below.

res.setHeader("cache-control", "no-cache"); for Http 1.1 based servers

res.setHeader("pragma", "no-cache"); for Http 1.0 based servers

\* Buffer is a temporary memory which stores the data for temporary period. Buffer is also called as cache.

\* for related information on Http request details, Http response details refer page nos 46 - 51

\* Every software and non-software objects life cycle (considering all the operations that are taken place from object birth to object death).

\* our servlet class object life cycle will be managed by Servlet Container

\* Event is an action performed / raised on the object.

\* servlet Container raises the following life cycle events in the life cycle of our servlet program. They are

1) Instantiation Event (raises when servlet Container creates our servlet class object)

2) RequestArrival Event (raises when browser window's servlet container takes the browser window generated request)

3) Destruction Event (raises when servlet Container is about to destroy our servlet class object)

\* when Container raises these events on our servlet class object it looks to call certain methods automatically so these methods are called as life cycle methods or Container callback methods. They are

init(servletConfig cg)

service (servletRequest req, servletResponse res)

destroy()

prototype of server life cycle methods

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

```
public void init(ServletConfig cg) throws ServletException
```

```
public void olestroy()
```

\*for one life cycle event only one life cycle method will be called by

Servlet Container calls init(), protected service (HttpServletRequest, HttpServletResponse), doXxx() methods are not life cycle methods.

\* servlet container calls init(-) life cycle method for instantiation event.

\* Servlet Container calls service( , ) life cycle method for Request-Arrival Event.

\* servletContainer calls destroy() lifecycle method for destruction Event.

\* The method that is called by underlying container automatically based on the event that is raised on the object is called Container callback methods.

\* Programmer never calls life cycle methods manually, they will be called by underlying container automatically.

\* Servlet API as supplied life cycle methods

(i) to allow programmer to place his choice logics in the execution of server programs.

(ii) to supply Servlet container created objects to servlet program (also for programmer as parameters of say lifecycle methods. (like request object, response object, servletConfig object and etc...))

\* while overriding super class method in sub class, the method can have same modifier or access modifier.

\* while overriding protected service() method of predefined HttpServlet class in its subclass we can take either protected or public modifiers for that method.

```
public class TestSrc extends HttpServlet
```

3

```
public void service (HttpServletRequest req, HttpServletResponse res) throws
```

1

here protected method

`ServletException; IOException`

—

is overridden

### service() method - 1

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

### service() method - 2

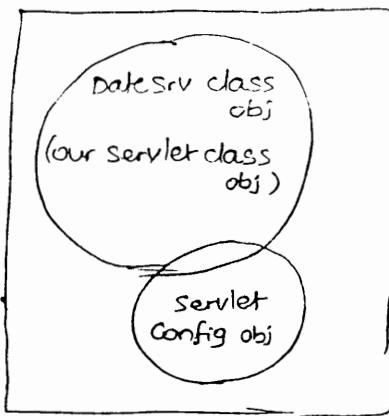
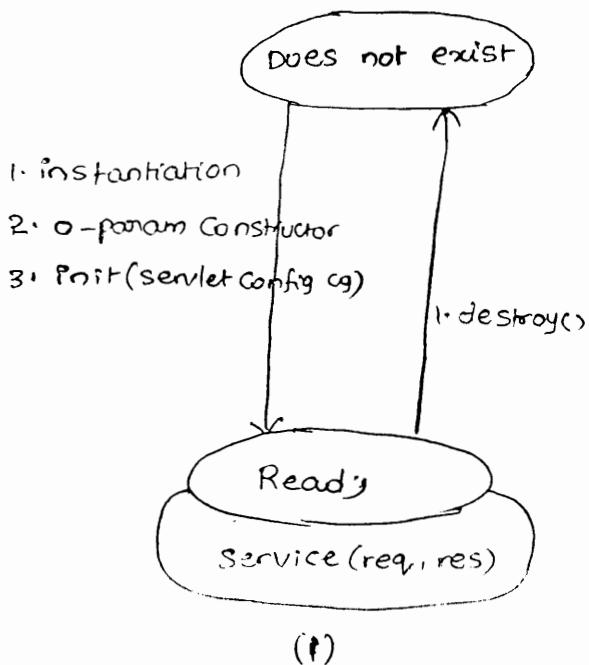
```
protected void service(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
```

\* servlet container is responsible to raise various life cycle events on our servlet class objects so logics placed in life cycle methods are not responsible to raise these events but they are responsible to process these events by executing programmers supplied logic.

Ex: code which is written in init() life cycle method never creates our servlet class object but programmer places this code has initialization logic to execute for instantiation event raised by servlet container by creating our servlet class object.

\* Life cycle method and its logic are not capable of raising life cycle events on the object but when underlying container raises life cycle events on the objects it automatically calls a relevant life cycle methods.

### Servlet life cycle Diagram :



(2)

(1) → servlet config object is right hand object to our servlet class object.  
it is one per servlet class object.

(2) → programmer uses this object to pass additional information to our servlet program and to gather information from our servlet program.

what happens when our servlet program gets 1<sup>st</sup> request from browser window?

(1) servlet container loads our servlet class from WEB-INF\classes folder of deployed web application.

(2) servlet container instantiates (object creation) our servlet class object as `Class.forName("DateSrv").newInstance();`

\* `Class.forName("DateSrv")` → loads our servlet class  
\* `newInstance()` → creates object for the loaded class DateSrv.

(3) During instantiation process the no-param constructor of our servlet class executes.

→ (4) Servlet Container calls `init(-)` life cycle method having `ServletConfig` object as argument value on our servlet class object.

→ (5) Servlet Container creates one `ServletConfig` object for our servlet class object.

\* (1) to (5) steps completes instantiation and initialization on our servlet class object.

(6) Servlet Container calls next life cycle method `service(-,-)` on our servlet class object. This will process the request and generated response goes to browser window as web page through web server.

what happens when our servlet program gets other than first request?

\* Servlet Container checks the availability of our servlet class object.

(1) if available, servlet container calls `service(-,-)` (public) on existing object of our servlet class to process the request and this response goes to browser window.

(2) if not available, servlet container performs all operations of 1<sup>st</sup> request.

## Example Servlet program to understand the life cycle of servlet

Adding this servlet program to MIMEAPP web application.

Step(1): Add LCTestSrv.java in WEB-INF/classes folder of deployed MIMEAPP web application.

//LCTestSrv.java

```
package P;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LCTestSrv extends HttpServlet
{
    public LCTestSrv()
    {
        System.out.println("LCTestSrv : o-param Constructor");
    }
    public void init(ServletConfig cg)
    {
        System.out.println("LCTestSrv : init(-)");
    }
    public void service(ServletRequest req, ServletResponse res) throws
        ServletException, IOException
    {
        System.out.println("LCTestSrv: service (-,-)");
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        //write business logic
        pw.println("Date and time is "+new java.util.Date().toString());
        //close stream object
        pw.close();
    } //service(-,-)
    public void destroy()
    {
        System.out.println("LCTestSrv: destroy()");
    }
} // class
```

\* Compile the servlet program source code.

D:\Tomcat 6.0\webapps\MIMEApp\WEB-INF\classes>javac -d . LCTestSrv.java

Step(2): Configure the above servlet program in web.xml having url pattern.

In web.xml (already some code is there it is added to that one)

```
<Servlet>
    <Servlet-name> abc4 </Servlet-name>
    <Servlet-class> p.LCTestSrv </Servlet-class>
</Servlet>
<Servlet-mapping>
    <Servlet-name> abc4 </Servlet-name>
    <url-pattern> /lct </url-pattern>
</Servlet-mapping>
```

Request URL to test App is

\* The servlet Program is there in package that must be specified in web.xml while configuring that servlet program.

Step(3): Start the server

\* once webServer is down all objects created and managed by that Server will be destroyed automatically (including our servlet class objects).

\* init(-) is a one time execution block of servlet program life cycle.

\* This method executes when Servlet Container raises instantiation event on our servlet class so the programmer generally keeps initialization logic like creating JDBC connection in this method.

\* service(-,-) method is repeatedly executing block of servlet program life cycle. Servlet Container calls this method for every request given to servlet program. So programmer generally keeps request processing and response generation logics in this method.

\* destroy() is the one time execution block of our servlet program life cycle. Servlet Container calls this method when it is about to destroy our servlet class object. So programmer generally keeps uninitialization logic like closing JDBC connection in this method.

when does servlet container creates our servlet class object?

- (A) when <load-on-startup> is not enabled on servlet program
- (B) when servlet program gets first request from client.
- (C) when servlet program gets first request after restarting web application.
- (D) when servlet program gets first request after reloading the web application.
- (E) when servlet program gets first request after restarting the server.
- (F) when servlet program gets first request after redeploying the web application.

when <load-on-startup> is enabled

- (A) The servlet container creates our servlet class object either during server startup or during the deployment of the web application.

Example code in web.xml

```
<servlet>
  < servlet-name> abc4 </servlet-name>
  < servlet-class> P-LCTestSrv </servlet-class>
  < load-on-startup> 5 </load-on-startup>
</servlet>                                <!--> load-on-startup> priority value
<servlet-mapping>
  < servlet-name> abc4 </servlet-name>
  < url-pattern>/lc </url-pattern>
</servlet-mapping>
```

when Servlet Container destroys our servlet class object through garbage collector?

- (A) when server is stopped or restarted.
- (B) when webapplication is stopped or reloaded.
- (C) when webapplication is undeployed.
- (D) when our servlet object sits idle continuously for long time.

NOTE: No special priority towards destroying our servlet class object even though load-on-startup is enabled on the servlets.

why should we enable load-on-startup on servlet program?

- the first request given to servlet goes to service() method only after the completion of instantiation and initialization process whereas other

than first request coming to servlet program go to service(--) method directly. Due to this the request processing and response generation time of first request will be little bit high when compare to other than first request.

To minimize first request request processing , response generation time and equalize that time with other than first request make servlet container to complete instantiation and initialization operations on servlet program before first request either during server start up or during deployment of web application by enabling <load-on-startup> on servlet program.

NOTE: Don't enable <load-on-startup> on every servlet program . Enable it only on those servlet programs of web application which will be guaranteedly requested by clients immediately after deploying the web applications like the servlet programs which generate home pages or main menus and etc..

\* When multiple servlet programs of web application are enabled with <load-on-startup> in which order the servlet container creates those servlet program objects will be decided based on their <load-on-startup> priority values.

\* High value indicates low priority . Low value indicates high priority

NOTE: <load-on-startup> with negative value is equal to not enabling <load-on-startup> on servlet .

#### Servlet prgs of WAI web application

<1-0-s>

Srv 1	1 (I)
Srv 2	3 (III)
Srv 3	2 (II)
Srv 4	10 (IV)

#### Servlet prgs of WA2 web application

<1-0-s>

Srv1	11 (III)
Srv2	0 (I)
Srv3	2 (II)
Srv4	-1 (Ignores <1-0-s>)

#### Servlet prgs of WA3 web application

<1-0-s>

Srv 1	4 }
Srv 2	4 }
Srv 3	4 }

Server decides the order

\* In Tomcat 5, 5.5 versions "0" is having least priority towards `<load-on-startup>` priority values where as in tomcat 6.0 "0" is having first priority or higher priority as shown above.

\* If you keep `<load-on-startup>` tag with out any priority value then that servlet program gets first / higher priority compare to other servlet programs whose `<load-on-startup>` tags are there with values.

Ex: `<load-on-startup> </load-on-startup>`

↓

This tag `<load-on-startup>` priority value is zero.

\* when no value is passed explicitly as priority value then the `<load-on-startup>` tag takes "0" as the priority value.

#### HTML to servlet Communication :

\* so far we have sent request to servlet program from browser window by typing request URL in the browser window. In this process to send data along with request we need to add query string to the request URL explicitly. But this work can be done only by technical people. The non technical end users like civil engineers, chemical engineers, kids can't do this work. So they need a graphical user interface to generate request with or without data. For that purpose we can use either HTML form page or hyperlink to generate request with or without data.

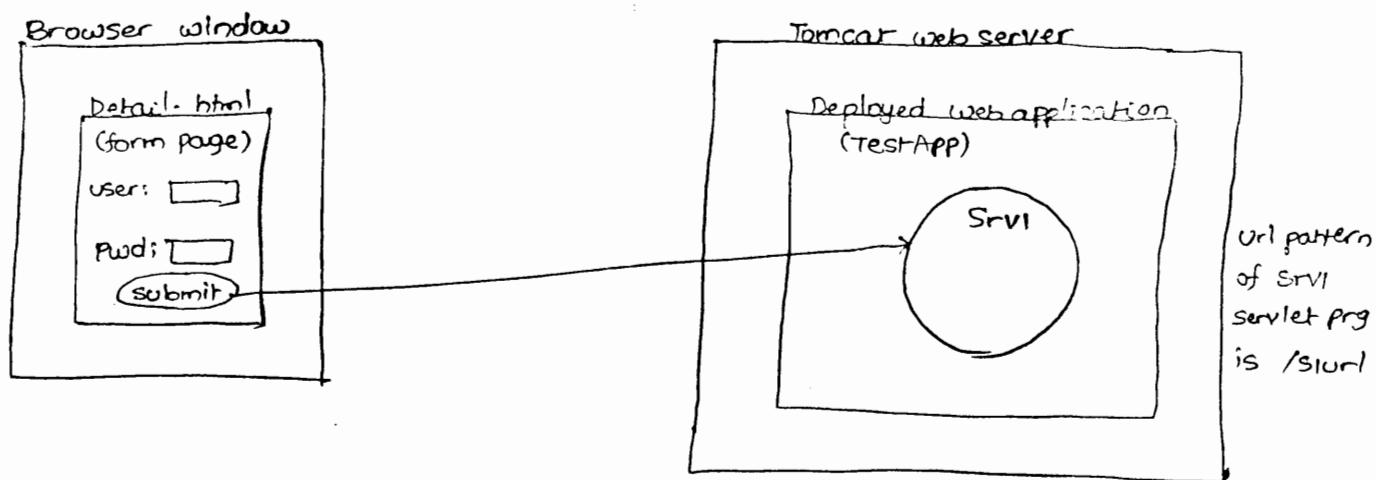
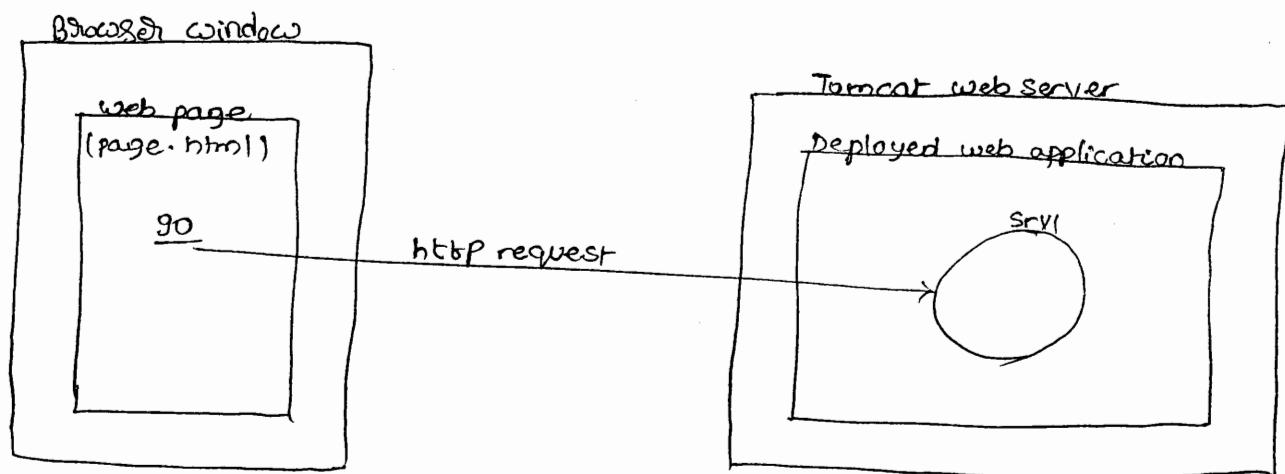


fig: Form page to servlet Communication

\* The form page generated request carries form data as request parameters along with request.

### Hyperlink based webpage to servlet communication

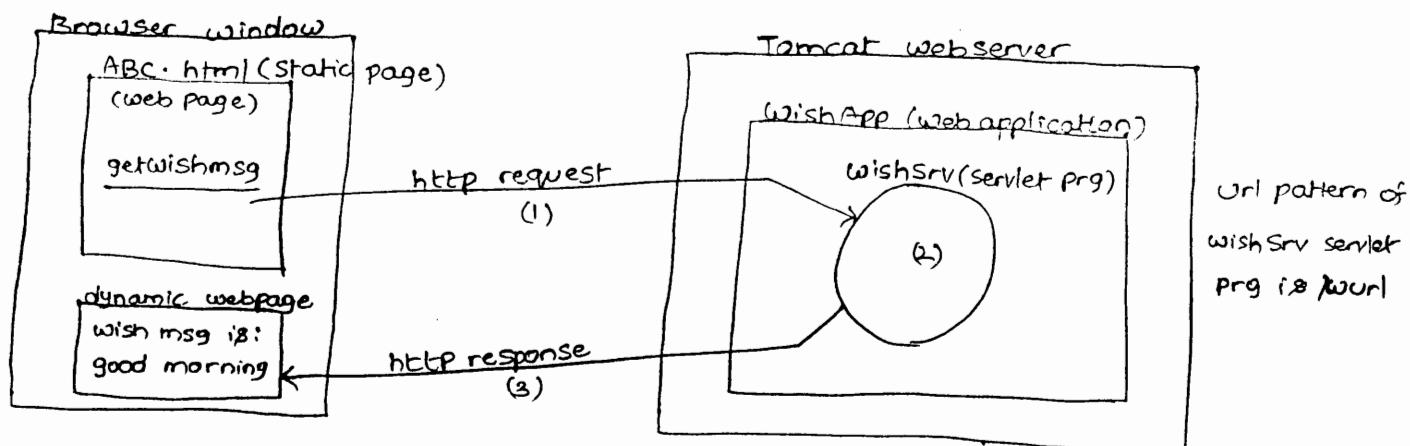


url pattern of Srv1 servlet  
Prg is \surl

\* Generally the hyperlink generated request is blank request that means it can not carry any data along with the request.

\* .html files of web application must be placed parallel to WEB-INF folder in deployment directory structure of web application. There is no need of configuring them in web.xml file.

### Example application (hyperlink based web page - servlet program Communication)



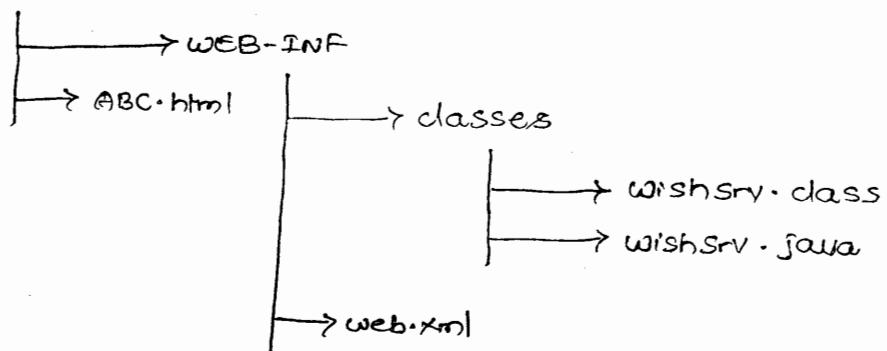
url pattern of  
wishesrv servlet  
Prg is \wurl

\* wishesrv servlet program generates the wish message based on the server machine's current time.

\* .html based web pages are static pages always, whereas servlet, JSP programs based web pages can be static pages or dynamic pages.

### Deployment Directory structure

Wish APP



\* place Servlet program request URL with URL pattern as the value of h-ref attribute.

↓  
in <a> tag to make hyper link generate request communicating with servlet program.

### ABC.html

```
<!-- web page having hyper links -->
<a href = "http://localhost:2020/wishApp/wurl">get wishmsg </a>
                                ↓
                                url pattern of wishsrv servlet program
```

### wishsrv.java

//wishesrv.java

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.\*; import java.util.\*;

public class wishesrv extends HttpServlet

{

public void service(HttpServletRequest req, HttpServletResponse res)

throws ServletException, IOException.

{

// set response Content type

res.setContentType("text/html");

```
//get printwriter obj  
PrintWriter pw = res.getWriter();  
//write request processing logic to generate wish message  
Calender cl = Calender.getInstance(); //gives current date  
and time  
//get current hour of the day  
int h = cl.get(Calender.HOUR_OF_DAY); //in 24hrs format  
//generate wish message  
if (h < 12)  
    pw.println("Good morning");  
else if (h < 16)  
    pw.println("Good afternoon");  
else if (h < 20)  
    pw.println("Good evening");  
else  
    pw.println("Good night");  
//close stream obj  
pw.close();  
} // service (-,-)
```

} // class

### web.xml

```
<web-app>  
    <servlet>  
        <servlet-name>abc </servlet-name>  
        <servlet-class>wishSrv</servlet-class>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>abc </servlet-name>  
        <url-pattern> /wurl </url-pattern>  
    </servlet-mapping>  
</web-app>
```

NOTE: do observe that html program is not configured in web.xml

### Request URL to test the application

http://localhost:2020/wishAPP/ABC.html

\* In absolute request URL all details will be specified including URL pattern or identification name of the target web resource program.

Ex: In ABC.html

http://localhost:2020/wishAPP/ABC.html

\* In relative request URL either URL pattern or identification name of target web resource program will be specified.

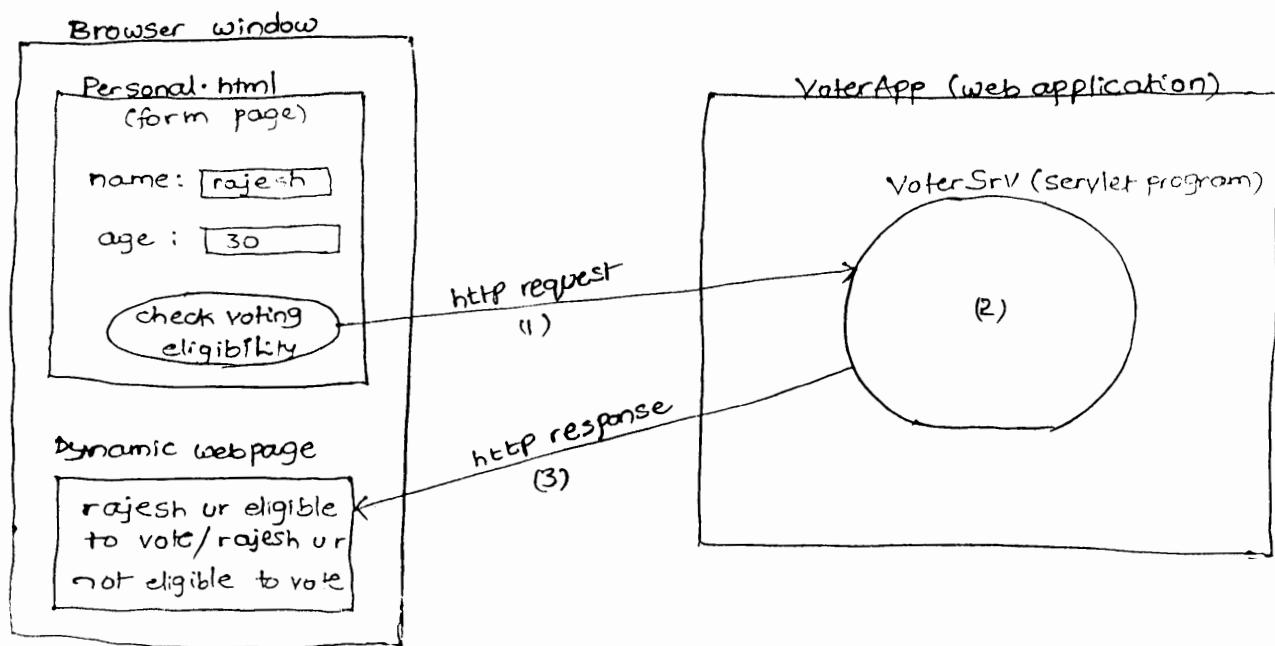
Ex: In ABC.html

<a href="wurl">getwishmsg </a>

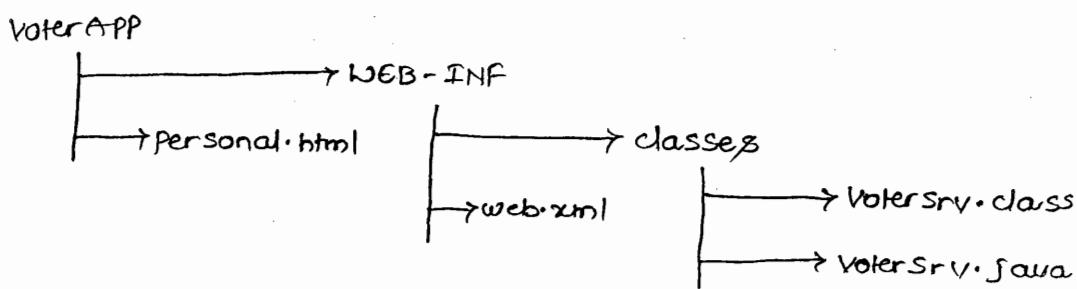
\* If source and destination web resource programs are there in the same web application then it is recommended to use relative URL. otherwise it is recommended to use absolute URL.

\* To make web applications as "WODA" applications it is recommended to use relative URLs.

### Example application on form page to servlet communication



## Deployment directory structure



## Personal.html

<!-- form page -->

```
<form action = "vturl" method = "get">
```

 URL pattern of VoterSrv servlet program (relative URL)

Name: <input type = "text" name = "pname"> <br>

Age : <input type="text" name="page"> name of the component

```
<input type="submit" value="Submit">
```

```
</form>
```

\* When form page is submitted the form component names will go to server as request parameter names (pname, page) and the form component values will go to server as request parameter values.

\* The "get" method based request generated by form page is recommended to process by using doGet(-,-) method in Servlet program. Similarly use doPost(-,-) method to process "post" method based request generated by form page.

\* Since `doXXX()` methods of servlet API are designed based on HTTP Standards so it is recommended to override `doXXX()` methods in our servlet programs to process the request.

Voter Srv. Java

## // VoterSrv.java

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

```

public class VoterSrv extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        //get PrintWriter obj
        PrintWriter pw = res.getWriter();
        //set response Content type
        res.setContentType("text/html");
        //read form data from form page as request parameter values
        String s1 = req.getParameter("Pname");
        String s2 = req.getParameter("page"); } Form Components acting
        //Convert age value to numeric value name
        int age = Integer.parseInt(s2.trim());
        //write logic to generate dynamic web page
        if (age >= 18)
            pw.println("<font color='green' size='4'>" + s1 + " u r eligible to
                        vote </font>");
        else
            pw.println("<font color='green' size='4'>" + s1 + " ur not eligible
                        to vote </font>");
        //add hyperlink to dynamic web page
        pw.println("<br><br><a href='personal.html'> home </a>"); } //doGet(-,-)
    } //class

```

### Web.xml

configure VoterSrv servlet program with "/vturl" URL pattern

Request URL to test the application

http://localhost:2020/voterApp/personal.html

\* To make our servlet program as flexible servlet program working for both "get", "post" http request methods we can use one of the following three approaches.

Approach(1) : By keeping service(-,-) method in our servlet program

```
Public class VoterSrv extends HttpServlet  
{  
    public void service(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException  
    {  
        ===== //request processing logic  
    }  
}
```

NOTE: The service(-,-) method of our servlet program can process both "get", "post" methods based request. But keeping request processing logic in service(-,-) method is not industry standard so try to keep request processing logic in doXXX methods.

Approach(2): By overriding both doGet(-,-), doPost(-,-) methods but keep request processing logic in one method and call that method from another method

```
Public class VoterSrv extends HttpServlet  
{  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException  
    {  
        ===== //request processing logic  
    }  
  
    public void doPost(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException  
    {  
        doGet(req, res);  
    }  
}
```

NOTE: Here servlet program can process both Get, Post method based request. Here request processing logic is not duplicated.

Approach (3): keep request processing logic in a user-defined method and call that method from both doGet( , ) , doPost( , ) methods

```
public class VoterSrv extends HttpServlet
{
    public void xyz(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        ===== // request processing logic
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        xyz(req, res);
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        xyz(req, res);
    }
}
```

\* Here the key benefits are same as approach (2).

\* Programmers generally prefer using approach (2) or approach (3) to make their servlet programs as flexible servlet programs against the http request methods "get", "post".

What happens if programmer calls destroy() method explicitly from the service( , ) method of servlet program?

- A) servlet container will not destroy our servlet class object but logic of destroy() method executes along with service( , ) method.  
When life cycle event is raised the servlet container calls life cycle method. Since the programmer has called life cycle method automatically.

the servlet container never raises the life cycle event.

what happens if init(-) method is called explicitly from the service(-,-) method of servlet program?

(A) servlet container never creates new object of our servlet class for the above call but logic of init(-) method executes along with service(-,-) method.

what is the difference between httpRequest method GET and POST ?

<u>GET</u>	<u>POST</u>
* Design to gather data from server by generating request.	* Design to send data to the server along with the request.
* GET send limited amount of data along with the request. (max of 256 kb)	* It can send unlimited amount of data along with the request.
* The form page <sup>generated</sup> query string will appear in browser's address bar so no data secrecy.	* The form page <sup>generated</sup> query string will not appear in browser's address bar. so data secrecy is available.
* Not suitable for file uploading operations.	* suitable.
* Can not send data in encrypted format.	* Can send.
+ use doGet() method or service(-,-) method to process the request	+ use doPost() method or service(-,-) method to process the request.
+ Get is not idempotent	+ Post is not idempotent.
* Default request method of HttpRequest.	+ is not default and should be applied explicitly.
* It is recommended to design form pages by having Post method.	
+ Hyperlink generated requests are get method based requests where as the form page can generate either get or post method based request.	
* Before completing the request processing of given request if client is	

allowed to generate next request and if web application is processing both the request then it is called as double posting problem or idempotent problem.

\* When form page submit button is clicked for multiple times this problem may raise. To prevent this problem take request method as post and process that request in doPost(..) method with additional logics. This indicates Post is not idempotent because it can prevent double posting by canceling all the requests.

\* The first web page of web application that comes automatically when request is given to web application is called as welcome page or home page of web application.

\* In java based web applications the html or jsp programs can be configured as welcome pages.

Ex: In web.xml of VoterApp web application

```
<welcome-file-list> must be taken as subtag of <web-app> tag  
  <welcome-file> Personal.html </welcome-file>  
  <welcome-file> Personal.htm </welcome-file>  
  <welcome-file> ABC.htm </welcome-file>  
  <welcome-file> ABC.jsp </welcome-file>  
</welcome-file-list>
```

only one file will become as welcome file here.

\* When no welcome file is explicitly configured the java web application looks to take either index.jsp or index.html as default welcome file.

\* If both are available the index.html will be taken as default welcome file.

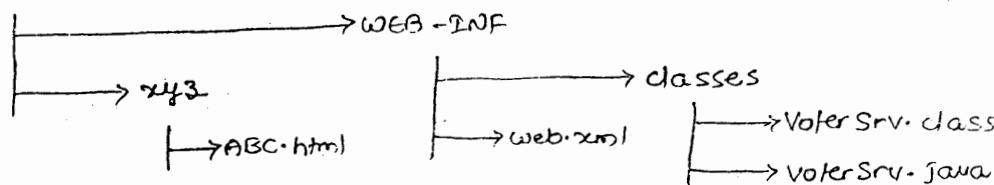
\* If multiple welcome files are configured then the web application picks up one welcome file based on the availability and configuration order.

\* If all explicitly configured welcome files are not available and if index.html or index.jsp files are available then web application runs without welcome file.

Note: we can not configure a servlet program as welcome file (in Tomcat 5.0, 6.0, 7.0)

\* The html files of web application can be placed in the sub directories of web root folder. while configuring these html files as welcome files we must specify that folder name.

### VoterApp



#### In web.xml

```
<welcome-file-list>
    <welcome-file> xyz\ABC.html </welcome-file>
</welcome-file-list>
```

\* Tomcat 6.0, 5.5 server allows to configure servlet program as welcome file by specifying its URL pattern.

#### In web.xml

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

└ URL pattern of servlet program

\* To make our servlet program related java class visible and accessible to servlet container, the java class must be taken as public class.

What happens if I place main(-) method in our servlet program?

- (A) servlet program execution will be taken care by servlet container through life cycle methods. Since main(-) is not the life cycle method of servlet program so it will not called by servlet container. only in stand alone applications we need main (-) method to begin the application execution. The container software managed programs or components will be executed through life cycle methods so main(-) method is not required in that components or programs.

How Servlet program is executing without main(-) method?

- (A) The stand alone application that will be executed by JVM directly needs to have main(-) method to begin the execution. Since servlet program is not stand alone application and it is a web resource program which will be executed by servlet container through life cycle methods so there is no need of main(-) method in servlet program.
- \* If already running java application wants to create certain other Java class object and wants to call methods of that class then that other class need not have main(-) method.
  - \* Servlet Container is a continuously running java application/software which creates our servlet class object and calls life cycle methods on that object for executing servlet program. So our servlet program need not to have main(-) method.

NOTE: Since we never give our servlet program to JVM directly for execution so there is no need of placing main(-) method in our servlet program.

\* Verifying the pattern and format of the form data before it is getting used in business logic as input values is called as form validation and logic written for this is called as form validation logic.

Ex: checking whether required components are filled up with values or not, checking whether age value is given as numeric value or not, checking whether email id is having @, . symbols or not.

What is the difference between form validation logic and business logic?

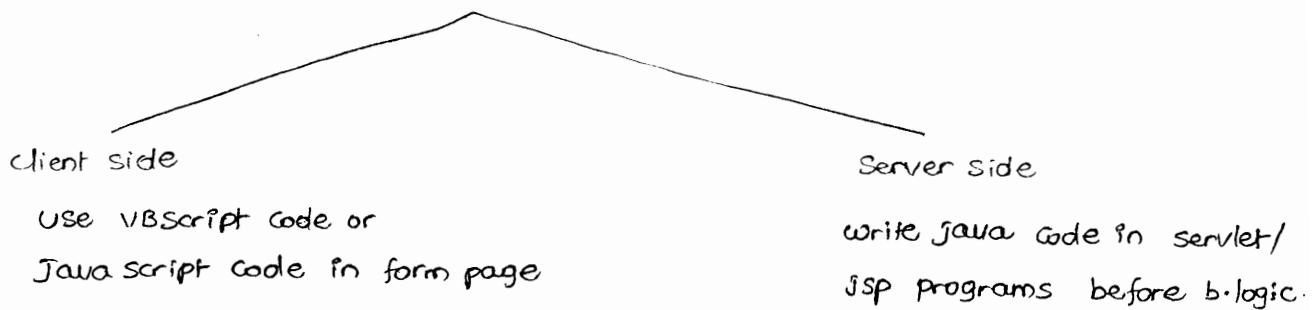
(A) Business logic uses the form data as input values and generates the results where as form validation logic verifies the pattern and format of the form data.

Ex(1): checking whether a, b values are typed or not and checking whether they are typed as numeric values or not comes under form validation.

\* performing  $c = a + b$  operation comes under business logic.

- Ex: \* checking whether user name and password values are typed or not comes under form validation.
- \* checking given username and password values against db s/w, username and password details comes under business logic.

### Form validations in web applications



- \* Since script code that is embed with form page executes by coming to browser window we can say client side form validations are good compare to server side form validations because the client side form validation reduces network round trips between browser window and web server.
- \* since there is a chance of disabling script code execution through browser settings it is recommended to write both client side and server side form validations so that there is a guarantee of performing server side form validations even though client side form validations are not done.
- \* To disable script code execution in IE

Tools → Internet Options → Security Tab → Custom Level → Scripting → Active Scripting → Disable

#### \* In netscap navigator

Edit → Preferences → Advanced → Scripts & Plugins → Enable java script for Navigator (deselect)

- \* return statement without value in any java method definition removes the control from current method definition and further statements in that method definition will not be executed.

\* For example application that performs both client and server side-form validations refer supplementary hand out of given on 13-10-2011

```
<form action="vturl" method="post" onSubmit="return validate(this)">
```

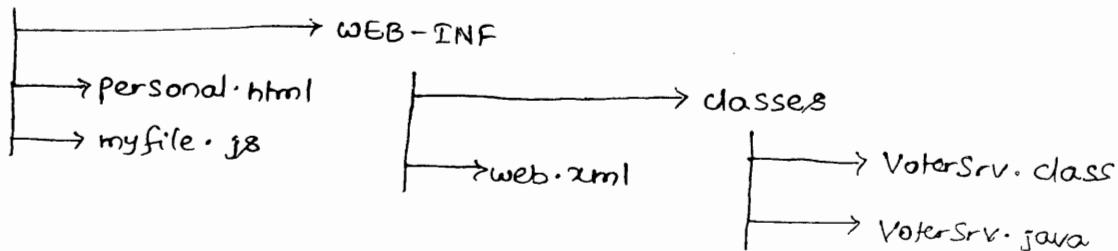
\* In the above statement the return key word takes the return value of validate function (true or false). If it is true the brows and sends to browser window.

\* If that value is true then browser window sends the request to the requested web resource program of web application. If the value is false then the browser window blocks/stops request going to server.  $\downarrow$  validation errors are there.

NOTE: on safe side the software industry writes both client side and server side form validations as shown in the hand out example.

\* To hide java script code from end users through view source option. Then and to make javascript code as reusable code for multiple html programs of web application then keep javascript code in .js file and link with html programs as shown below.

### VoterApp



#### Personal.html

```
<!-- form page -->
<html>
  <head>
    <script language="JavaScript" src="myfile.js">
    </script>
  </head>
  <form action="vturl" method="post" onSubmit="return validate(this)">
    <input type="text" name="username" />
    <input type="password" name="password" />
    <input type="submit" value="Login" />
  </form>
```

## myfile.js

```
function validate (frm)
{
    ===== //refer hand out code
} //validate()
```

\* The code that can not be executed independently and it must be embedded along with other technology based program is called as script code.

\* Javascript code can not be executed independently and it must be embedded with HTML program for execution. so the Javascript code is called as script code.

### what is the difference between java and javascript?

NOTE: Javascript is no way related with java

#### Java

\* It is a programming language to develop all kinds of applications.

\* Java application code can be executed independently.

\* Needs JRE/JVM for execution.

\* It is object oriented language.

#### javascript

\* It is a scripting language given for form validations and for other operations.

\* JavaScript code must be embedded with html code for execution.

\* Needs JavaScript engine for execution.

\* It is object based language.

Public class Test

{

Public void x()
 {

y();
 }

Public void y()
 {

=
 }

}

```
public class Demo extends Test
{
    public void y()
    {
        (d)
    }
}
```

```
Demo d = new Demo();
d.x(); (a)
```

### understanding init() vs init(ServletConfig cg) methods

\* init(ServletConfig cg) is lifecycle method of servlet program and init() is not lifecycle method and it is convenience method given to programmer to place initialisation logic of servlet program.

#### Scenario (1) :

##### GenericServlet.java (Pre-defined class)

```
public abstract class GenericServlet implements Servlet
{
    ServletConfig cg;
    public void init(ServletConfig cg)
    {
        this.cg = cg; //initialization logic of ServletConfig obj
        init();
    }
    public void init() //empty method
    {
    }
    public ServletConfig getServletConfig() { return cg; }
    other methods
}
```

NOTE: In predefined HttpServlet class there are no init() methods.  
(1) (2) (3)

##### TestSrv.java (Our servlet program)

```
public class TestSrv extends GenericServlet / HttpServlet
{
    public void init()
    {
        (5)
        Programmer choice initialization logic like creating JDBC
        Connection obj
    }
}
```

```
public void service(SReq req, SRes res) throws SE, IOE
```

```
{  
    (6)  
    ===== request processing logic  
}
```

\* When servlet container raises instantiation event by creating our servlet class object then container calls init(-) method but not init() method. So init(-) method is called as lifecycle method and init() method is not lifecycle method.

\* With respect to the above code

- (1) End user gives first request to our servlet program.
- (2) Servlet Container creates our servlet class object (TestSrv) and also creates one ServletConfig object.
- (3) Servlet Container calls init(-) method as lifecycle method by keeping ServletConfig object as argument value, on our servlet class object.
- (4) Since init(-) is not available in our servlet program the super class (Predefined GenericServlet class) init(-) method executes.
- (5) The super class init(-) method initializes the received ServletConfig object in GenericServlet class and calls init() method. Since init() method is available in our servlet program that will be executed. Here programmer places his servlet program related initialization logic.
- (6) Servlet Container calls next lifecycle method called public void service(-,-) on our servlet class object.

NOTE: In scenario (1) programmer need not to initialise the ServletConfig object explicitly in his servlet program. More ever we can call getServletConfig() method to get access to servletConfig object from any lifecycle method of his servlet program.

### Scenario (2)

#### GenericServlet.java

```
===== // Same as above code in scenario (1)
```

```

public abstract class GenericServlet implements Servlet
{
    ServletConfig cg;
    public void init(ServletConfig cg)
    {
        this.cg = cg;
        init();
    }
    public void init()
    {
        // Other methods
    }
}

```

### TestSrv.java (our servlet prg)

```

public class TestSrv extends GenericServlet/HttpServlet
{
    ServletConfig cg;
    public void init(ServletConfig cg)
    {
        (4) this.cg = cg;
        // Programmer choice initialization logic like creating
        // JDBC Connection obj
    }
    public void service(SReq req, SRes res) throws SE, IOE
    {
        (5) // request processing logic
    }
}

```

\* Here (1), (2), (3) are same as scenario (1)

(4) → init() method of Servlet program executes having initialization logic

(5) → same as step(6) of scenario (1).

NOTE (1): In scenario (2) the programmer must explicitly initialize ServletConfig object from init() method to give access to ServletConfig

object from other lifecycle methods (refer above). If programmer forgets to do this ServletConfig object initialization then the service() & destroy() lifecycle methods can not get access to ServletConfig object.

NOTE(2): In scenario (2) control did not pass on to init() method of super class (pre defined GenericServlet class) so our servlet program can not call getServletConfig() method to get ServletConfig object.

### Scenario (3)

#### GenericServlet.java (pre defined servlet)

```
Public class GenericServlet implements Servlet
{
    ServletConfig cg;

    Public void init(ServletConfig cg)
    {
        this.cg = cg; (6)
        init();
    }

    Public void init()
    {
        (7)
    }

    Public ServletConfig getServletConfig()
    {
        return cg;
    }
}

==== //Other methods
```

#### TestSrv.java (our servlet program)

```
(1) (2) (3)
Public class TestSrv extends HttpServlet/GenericServlet
{
    Public void init(ServletConfig cg)
    {
        (4)
        ===== // Programmer choice initialization logic like create jdbc
               connection
        super.init(cg); (5)
    }

    Public void service(ServletRequest req, ServletResponse res) throws SE, IOE
    {
        (6)
        ===== //request processing logic
    }
}
```

- (1), (2), (3) steps of Scenario (3) are same as (1), (2), (3) steps of scenario (1)
- (4) init() method of our servlet program executes.
- (5) Because of super.init(cg) method call the super class (GenericServlet) init() method executes.
- (6) init() method of GenericServlet class initializes the ServletConfig object  
Contains servletConfig initialization logic
- (7) Since init() method is not available in our servlet init() method  
of GenericServlet class will execute.
- (8) same as step (6) of scenario (1)

NOTE: In scenario (3)

- (i) Programmer should call super.init(cg) method in the definition of init() method of our servlet program and he can use getServletConfig() method call to get ServletConfig object.
- (ii) If programmer forgets to call super.init(cg) method the service(), destroy() lifecycle methods can not get access to ServletConfig object even after calling getServletConfig() method.
- \* The best scenario to keep init() method in our servlet program is scenario (1) (overriding init() method).
- \* Always give chance to execute init() method of pre defined GenericServlet class during our servlet class instantiation and initialization process. Because this method contains servletConfig initialization logic and makes programmer not performing that work explicitly.

why Servlet API has given init() method when it is not lifecycle method?

- Refer the 4 paragraphs of init() vs init(ServletConfig cg) discussion of page no. 55 of booklet.
- 
- A good servlet programmer always overrides init() method in his servlet program instead of init() method (follow scenario(1)).
- Self class methods and super class public/protected methods in sub class can be called without object.

In any life cycle method of servlet program

ServletConfig cg = getServletConfig(); //gives access to ServletConfig obj  
↓  
It is public method of GenericServlet class so it can be called without object in our servlet programs.

understanding two service(-,-) methods and seven doXXX(-,-) methods of predefined HttpServlet class

\* we can use one of the two service(-,-) methods or one of the seven doXXX(-,-) methods to place request processing logic in our servlet program but only public service(-,-) method is life cycle method of servlet program.

HttpServlet.java (predefined class)

```
public abstract class HttpServlet extends GenericServlet
{
    public void service (ServletRequest req, ServletResponse res) throws SE, IOE
    {
        // (4)
        // type casting
        HttpServletRequest hreq = (HttpServletRequest)req;
        HttpServletResponse hres = (HttpServletResponse)res;
        // calls protected service(-,-) method
        service(hreq, hres);
    }

    // (5)
    // protected service(-,-) / service(-,-) method2
    protected void service(HttpServletRequest hreq, HttpServletResponse hres)
        throws SE, IOE
    {
        // reads the request method of given request
        String method = req.getMethod();
        // calls one of 7 doXXX(-,-) methods based req method
        if (method.equals("GET"))
            doGet(hreq, hres);
        else if (method.equals("POST"))
            doPost(hreq, hres);
        else if (method.equals("head"))
            doHead(hreq, hres);
    }
}
```

```

        }

protected void doGet(HttpServletRequest req, HttpServletResponse res)
                     throws SE, IOE

{

    ===== logic send 405 error response to browser window

}

protected void doPost(HttpServletRequest req, HttpServletResponse res)
                     throws SE, IOE

{

    ===== logic send 405 error response to browser window

}

===== // implementation of other doXXX(-,-) methods sending 405
       error response to browser window.

```

3//class

#### TestSrv.java (our Servlet program)

```

(1) (2) (3)
public class TestSrv extends HttpServlet

{

    public void doGet(HttpServletRequest req, HttpServletResponse res)

    {

        (6)
        ===== our request processing logic
    }
}

```

with respect to above code

- (1) client gives request to our servlet program having "get" method.
- (2) servlet container creates (or) locates our servlet class object. If created servlet container also completes the initialization process.
- (3) The servlet container creates `ServletRequest`, `ServletResponse` objects for current request and calls the life cycle method `public service(-,-)` on our servlet

class object. keeping ServletRequest, ServletResponse objects as argument values.

(4) since public service(-,-) method is not available in our servlet program, the super class public service(-,-) method executes (Predefined HttpServlet class). This method converts ServletRequest object to HttpServletRequest object, ServletResponse object to HttpServletResponse object and calls protected service(-,-) method having those objects as arguments.

(5) since protected service(-,-) method is not available in our servlet program, the protected service(-,-) method of super class (predefined HttpServlet class) executes. This protected service(-,-) method calls doGet() method based on "get" method of given request.

(6) since doGet(-,-) method is available in our servlet program that will be executed and generated response goes to browser window.

NOTE: Don't let doXXX(-,-) methods of predefined HttpServlet getting executed directly or indirectly because they send "405" error response to browser window indicating that our servlet is incomplete servlet program towards processing request.

\* When request arrived event is raised (means request comes to container) servlet container calls public service(-,-) method on our servlet class object and it never calls protected service(-,-) method, doXXX(-,-) methods for that event. So only public service(-,-) method is called as servlet life cycle method, protected service(-,-) and doXXX(-,-) methods are called convenience methods given to programmer to keep request processing logic in our servlet programs.

\* For understanding flow of execution in our servlet programs towards the request processing refer the six scenarios given in page no. 56 of booklet.

\* Even though public service(-,-) method is lifecycle method but the good programmer always prefers overriding doXXX(-,-) methods in his servlet

program to place request processing logics. Because these methods are given based on the http protocol's standard. Moreover the super class (pre-defined HttpServlet class) version of this methods can generate appropriate error messages like 405 when those methods are really not overridden in our servlet program.

- \* most of the programmers in real world overrides both doGet(--) and doPost(--) methods in servlet program, but they place request processing logic only in one method and calls that method from another method.
- \* In java abstract class can have only abstract methods (or) only concrete methods (or) mix of both.

In predefined HttpServlet class when all methods are given as Concrete methods. why that class itself is given as abstract class?

(a) Refer the last 3 paragraphs of page no. 56 of booklet.

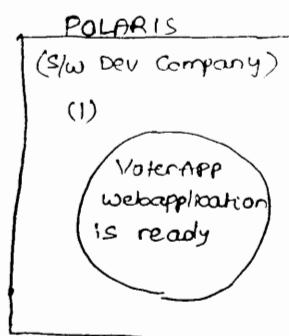
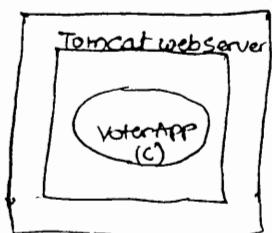
- \* Every website that is hosted in the internet will be having one domain name like www.yahoo.com, www.gmail.com and etc..
- \* These domain names along with home page request URLs will be registered in DNS register registering.

Procedure to host web application on to the internet having domain name

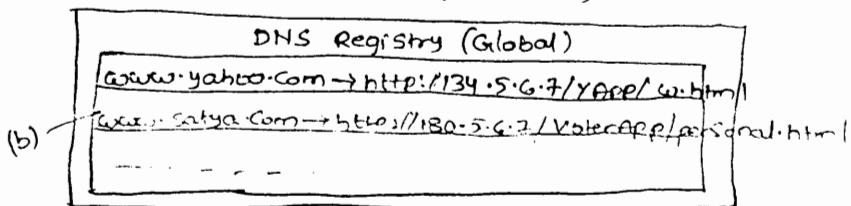
(2) Purchase domain name from ISP  
like www.satya.com

(3) Purchase space in websiterverapplication  
Server maintained by ISP in static  
IP address based machine.

(4) ISP machine (Ip Address: 180.3.5.7)

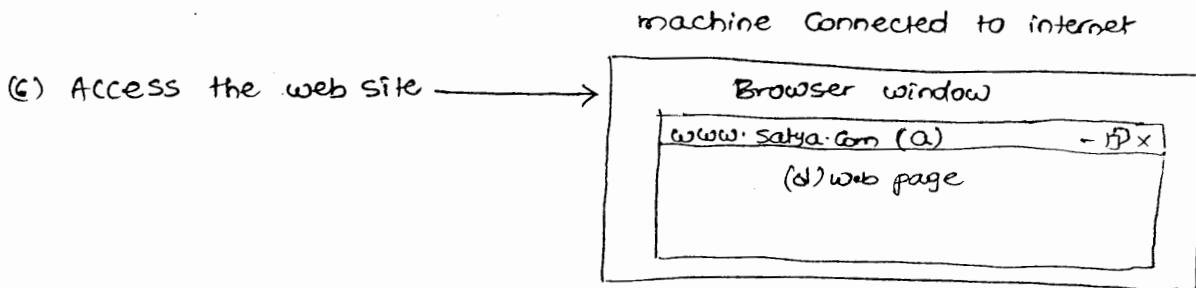


machine (Static IP address)



(4) move VoterAPP app the server of ISP machine

(5) register VoterAPP app in DNS registry.



\* Registering web application in DNS registry is nothing but keeping domain name of web application along with home page URL.

\* The ISP machine where ever web application hosted will be having static / fixed IP address.

\* Once we purchase space from the server of ISP machine they supply FTP application along with user name and password <sup>to us</sup> and we can use that FTP application to interact with our space of ISP machine from any place so using this we host or move web applications to the server of ISP machine.

#### Flow of accessing

(a) End user types domain name as URL in browser windows address bar.

(b) Through internet network the request goes to DNS registry, gathers home page request URL of given domain name (like satya.com) from DNS registry.

(c) Based on home page request URL the web resource program of hosted web application in tomcat server of ISP machine will be executed.

(d) The web resource program generates one web page to browser window.

\* We can add or remove web resource programs in the hosted website of ISP server from our local machine through FTP application. (This is nothing but updating web pages content after hosting).

## http request methods

GET  
POST  
HEAD  
DELETE  
PUT  
TRACE  
OPTIONS

\* The regularly used two request methods of real world programming are get, post.

### GET:

\* Given to gather / to get more data from the server.

\* The response of this method based request contains both headers and body. Head

### Head:

\* Same as get but this method based request generated response contains only response headers.

\* This method is useful to check the availability of web resource programs.

NOTE: Get() method is useful to gather data from server by sending limited amount of data (upto 256 kb) from the request.

### post:

\* Design to send unlimited amount of data along with the request.

### put:

\* Useful to add new web resource program in web application from client.

### delete

\* Useful to remove web resource program of web application from client.

NOTE: Put(), delete are useful in the development of FTP applications.

### options

\* Server determines using which request methods the current web resource program can be requested.

NOTE: If our server program contains doGet() method overriding then the options request method based request given to that servlet program

returns get, Head, Trace, options.

### trace

\* A trace request method based request sends the flow of execution details of certain web resource program like servlet. So these details can be used for debugging operations.

### List of form Components in form page

(1) TextBox

(2) Password Box

(3) Select box / ComboBox

(4) List box

(5) TextArea

(6) Radio Buttons

(7) CheckBox

(8) Button (Standard button, submit button, reset button)

(9) File uploading Component

### TextBox

#### In form page

Name: <input type="text" name="Pname">

#### In servlet program (to read comp value)

```
String s1 = req.getParameter("Pname");
```

### Password box

#### In form page

Age: <input type="password" name="page">

#### In servlet program

```
String s1 = req.getParameter("page");
```

### TextArea

#### In form page

Address: <textarea name="taddress" rows="4" cols="5">

enter ur address

</textarea>

In servlet program (to read component value)

```
String s1 = req.getParameter("taddress");
```

Select Box / Combo Box (allow us to select one item at a time)

Qualification : <select name="q1fy">

```
<option value="engg">B.E/B.Tech</option>
<option value="medico">MBBS</option>
<option value="arts">B.A</option>
</select>
```

In servlet program

```
String s1 = req.getParameter("q1fy");
```

\* while working with Select Box the selected item will not come to server as request parameter value, the value available in the value attribute of option tag for selected item will come to server as request parameter value.

Ex: From the above select box if BE/B.Tech is chosen then the value engg will go to server as request parameter value.

List box (allows us to select multiple items at a time)

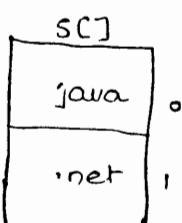
In form page

```
Courses: <select name="crs" multiple>
<option value="java"> JAVA PKG </option>
<option value=".net"> .NET PKG </option>
<option value="Oracle"> Oracle PKG </option>
</select>
```

In servlet program

```
String s[] = req.getParameterValues("crs");
```

\* If JAVA PKG, .NET PKG items are chosen then s[] holds java, .net as elements values.



## Radio buttons

\* By giving same name for multiple radio buttons we can group them into single unit. So only one radio button can be selected at a time.

### In form page

Gender: <input type="radio" name="g" value="m">Male    <input type="radio" name="g" value="F"> Female

### In Servlet Program

```
String s1 = req.getParameter("g"); // gives m when male radio is selected  
// gives F when female radio is selected.
```

## checkboxes

\* When multiple checkboxes are grouped into single unit by giving same name then we can select multiple checkboxes at a time.

### In form page

Hobbies: <input type="checkbox" name="ch1" value="read"> Reading

<input type="checkbox" name="ch1" value="sleep"> sleeping

<input type="checkbox" name="ch1" value="roam"> roaming

### In Servlet Program

```
String s[] = req.getParameterValues("ch1");
```

\* If reading, roaming checkboxes are selected then the s[] holds read, roam values.

## NetBeans

Type: IDE for Java environment

Version: 6.7.1 (compatible with jdk1.6)

Vendor: Sun micro system

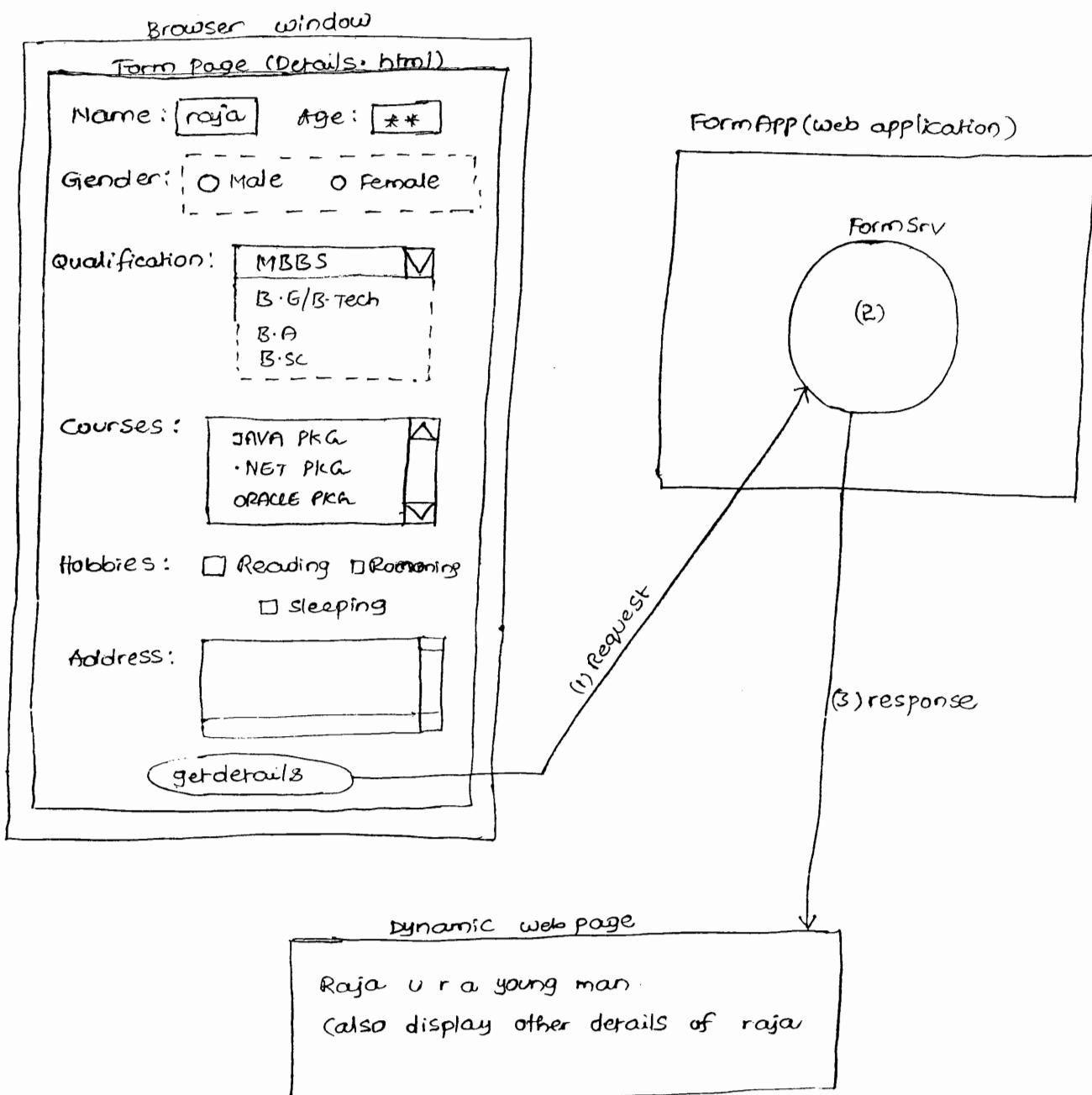
gives GlassFish 2.x as Built-in Server

open source S/W

to download S/W: [www.netbeans.org](http://www.netbeans.org)

for docs: [www.netbeans.org](http://www.netbeans.org)

\* It is always recommended to design the form components of form page as HTML table rows and column values. This allows us to align the form page easily.



Procedure to develop and execute above application by using NetBeans IDE

Step(1): create java web project having name FormApp.

File menu → new project → java web → web application → next →

Project Name:

→ next → server:  → next → Finish

Step(2): Add form page to the project (Details.html)

Right click on web pages folder → new → HTML → HTML File Name:   
 → Finish

```

<form action="furl" method="get">
    ↓
    url pattern of FormSrv servlet program

    <table border="1">

        <tr>
            <td> Name: </td>
            <td> <input type="text" name="Pname" ></td>
        </tr>

        <tr>
            <td> Age: </td>
            <td> <input type="password" name="Page" ></td>
        </tr>

        <tr>
            <td> Gender: </td>
            <td>
                <input type="radio" name="g" value="m" checked
                <input type="radio" name="g" value="F"
            </td>
        </tr>

        <tr>
            <td> Qualification </td>
            <td>
                <select name="qlfy">
                    <option value="java" > B.G/B.Tech
                    <option value="Net" > M.B.B.S </option>
                    <option value="c++" > B.A </option>
                    <option value="Science" > B.Sc </option>
                </select>
            </td>
        </tr>

        <tr>
            <td> Courses </td>
            <td>
                <select name="crs" multiple>
                    <option value="java" > JAVA PKG </option>
                    <option value=".Net" > .NET PKG </option>
                    <option value="oracle" > Oracle PKG </option>
                </select>
            </td>
        </tr>
    
```

```

<tr>
    <td> Hobbies </td>
    <td>
        <input type="checkbox" name="chi" value="read" checked />
        reading &nbsp; &nbsp;
        <input type="checkbox" name="chi" value="roam" /> Roaming
        &nbsp; &nbsp;
        <input type="checkbox" name="chi" value="sleep" /> sleeping
    </td>
</tr>
<tr>
    <td> Address </td>
    <td>
        <text area name="address" rows="4" cols="20>
        enter address
        </text area>
    </td>
</tr>
<tr>
    <td colspan="2" > <input type="submit" value="getdetails" /> </td>
</tr>

```

↓  
merges two cells into single cell

```

</table>
</form>

```

Step(3): Add FormSrv servlet to source packages folder of project.

Right click on Source Packages folder → new → servlet →

class name : FormSrv → next →

url pattern : /url → finish

NOTE: The above IDE generated servlet program gives `processRequest(..)` method to programmer for keeping request processing logic and this method will be called internally from `doGet(..)`, `doPost(..)` methods like approach (3) of developing our servlet program as flexible servlet program.  
(Refer approach (3))

step(4): keep the following request processing logic in processRequest(..) method of above servlet program.

```
{  
    //general settings  
    PrintWriter pw = res.getWriter();  
    res.setContentType("text/html");  
    //read from form page  
    String name = req.getParameter("Pname");  
    int age = Integer.parseInt(req.getParameter('page'));  
    String gen = req.getParameter("g");  
    String qlfy = req.getParameter("qlfy");  
    String crs[] = req.getParameterValues("crs");  
    String chl[] = req.getParameterValues("chl");  
    String address = req.getParameter("taddress");  
    if (gen.equals("F"))  
    {  
        if (age <= 5)  
            pw.println(name + " u r baby girl");  
        else if (age <= 12)  
            pw.println(name + " u r child girl");  
        else if (age <= 19)  
            pw.println(name + " u r teenage girl");  
        else if (age <= 30)  
            pw.println(name + " u r a young woman");  
        else if (age <= 45)  
            pw.println(name + " u r middle age woman");  
        else  
            pw.println(name + " u r old lady");  
    }  
    else if (gen.equals("M"))  
    {  
        if (age <= 5)  
            pw.println(name + " u r baby boy");  
    }  
}
```

```

//close stream obj
pw.close();

// print form data
pw.println("<br> name = " + name);
pw.println("<br> age = " + age);
pw.println("<br> Address = " + address);
pw.println("<br> Gender = " + gen);
pw.println("<br> Qualification = " + qlfy);
pw.println("<br> Courses = ");

for (int i=0; i< crs.length; ++i)
{
    pw.println(crs[i] + "...");
}

pw.println("<br> hobbies = ");
for (int i=0; i< hb.length; ++i)
{
    pw.println(hb[i] + "...");
}

//close stream obj
pw.close();
}

```

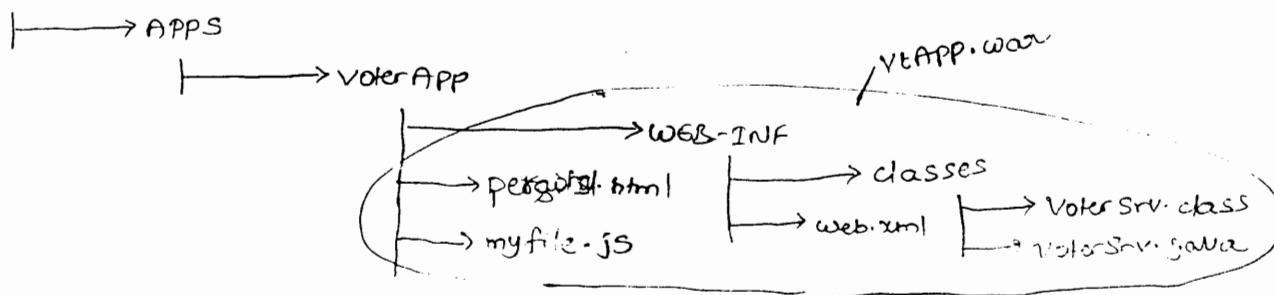
Step(4) : Run the project

Right click on FormApp Project → Run

\* we can prepare WAR file on deployment directory structure of web application. Each war file represents one web application.

Procedure to prepare war file

e:\



To prepare war file.

e:\APP\VoteApp> jar cf vtapp.war .

\* Gives war file by combining everything of VoteApp folder.

\* There are 3 ways to deploy web applications in servers.

- (1) Hard deployment (Copy war or directory of Web application to a fixed installation folder of server software like Tomcat\\_home\webapps folder).
- (2) Console deployment (use admin console window for deployment)
- (3) Tool based deployment (Deployment using tools like Ant, maven, IDEs)

\* (3) is recommended process.

### Procedure to deploy webapplication in Tomcat server through console deployment process

Step(1) : prepare war file (vtapp.war) as shown above

Step(2) : Launch Tomcat web application manager window

start tomcat Server → launch home page → Tomcat manager → submit

user name password

Step(3) : Deploy the war file

Tomcat web application manager window → Select WAR file to upload →

browse → select the above vtapp.war file → Deploy

\* Test the application by using the following request URL

http://localhost:2020/vtapp/personal.html

↓  
war file name as context path

NOTE : In most of the servers when war file is deployed, its file name automatically becomes context path of the web application.

\* To perform hard deployment of web application in tomcat server.

copy VoteApp folder (or) vtapp.war file to tomcat\\_home\webapps folder.

## weblogic

Type : Application Server S/w

Vendor: BEA Systems (Oracle Corp)

Version: 10.3 (Comparable with JDK 6)

Commercial S/w

default port no: 7001

JAR file that represents whole See ports API: weblogic.jar

To download software: www.oracle.com or www.commerce.bea.com

for docs: www.edocs.bea.com or www.oracle.com

Allows to create domains. Each domain act as one application server.

\* If multiple projects of a company are using

then weblogic S/w will be installed only once in a common machine  
but for multiple projects multiple domains will be created in that S/w.

Procedure to create user define domain server in weblogic 10.3

Start → Programs → Oracle WebLogic → Quick Start → Get Started with  
WebLogic Server → Create New WebLogic Domain → Next → Generate a  
Domain . . . → Next →

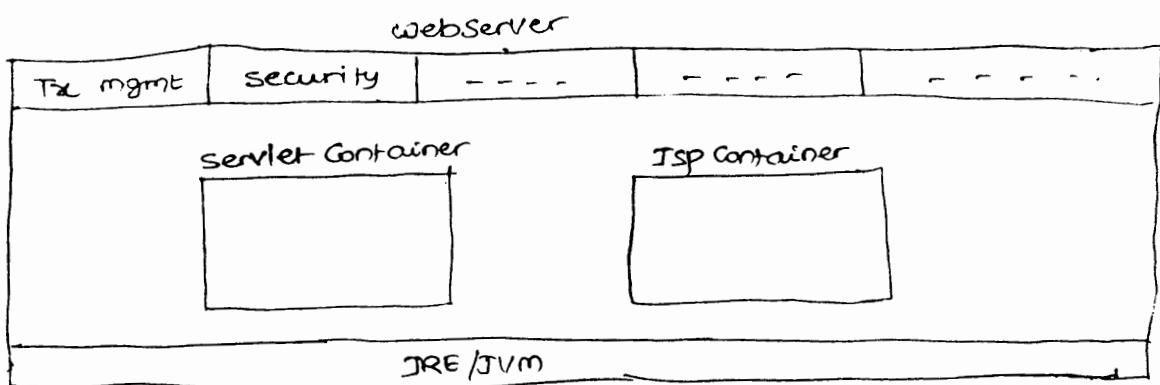
Domain name: Adv.javaBatchDomain

→ Next → User name: javaboss Password: javaboss1 Conform

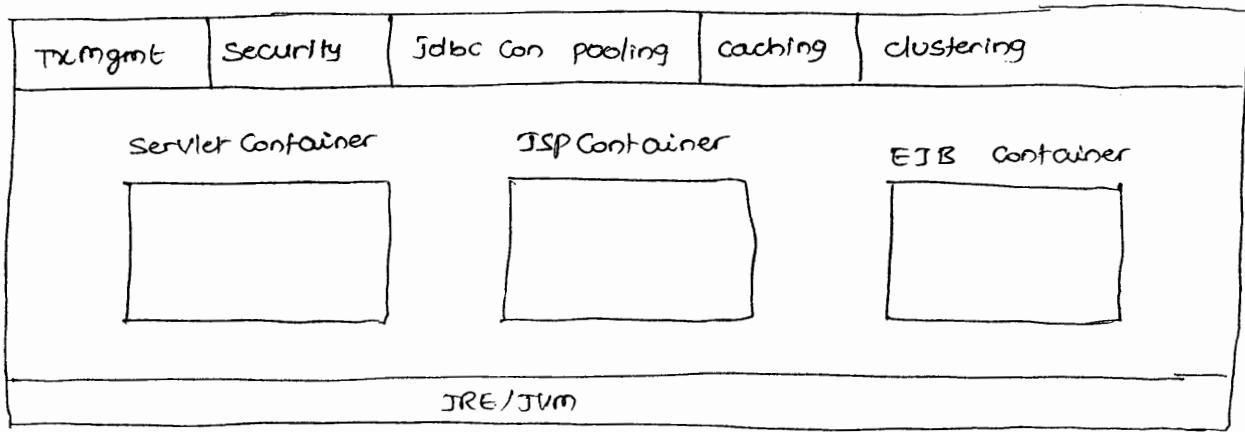
User password: javaboss1 → Next → Select Administration Server →

Next → Listen port: 7070 → Next → Create.

Application Server = Web Server + EJB Container + middle ware services



## Application Server



- \* EJB Container is required to manage and execute EJB component.
- \* war file → web application archive (represents web application)
- \* jar file → Java archive (represents Java API/Zip file)
- \* ear file → enterprise application archive (jar file + war file + jar file + war file + ...)
- \* rar file → resource adaptor archive (represents a JEE application interacting with Siebel/SAP S/W)

What is the difference between webserver and application server?

### webserver

(1) Allows to deploy and execute web applications.

(2) Developed based on servlet, JSP api specifications.

(3) Gives Servlet Container, JSP Container.

(4) Doesn't allow to create domain

(5) Allows only Http protocol request

(6) Gives minimum no. of middleware services.

### application server

(1) Allows to deploy and execute web applications, EJB Components, Enterprise applications and resource adapter apps.

(2) developed based on all JEE api specifications. (Servlet 3, JSP, EJB, JMS etc.)

(3) Gives both Servlet Container, JSP Container and EJB Container.

(4) Allows to create domains.

(5) Allows both Http and non Http (IIOP, T3 and etc) protocol based request.

(6) Gives more no. of middleware services.

- (7) suitable for small scale and medium scale web applications.
- (7) suitable for large scale web applications and for JEE applications.
- (8) Recognizes .war file as application
- (8) Recognizes .war, .ear, .jar, .rar files as applications.
- (9) Ex: IWS, Tomcat, Resin and etc.
- (9) Ex: weblogic, websphere, JBOSS, GlassFish and etc.

Procedure to perform the console deployment of web application Adv. batch Adv.javaBatchDomain

Step(1): Prepare war file representing your web application like VtApp.war

Step(2): Start Adv.javaBatchDomain server of web logic. refer previous class

start → Programs → Oracle weblogic → user projects → Adv.javaBatchDomain  
 → start admin server for web logic.

Step(3): open administration console of the above domain server.

open browser window → type below URL.

http://localhost:7070/console

username: jboss

password: jboss1

Step(4): Deploy the web application through admin console.

Admin console → environment → Deployment → install → upload your files → deployment archive → Browse and select VtApp.war file → next → next → next → next → finish → save

Step(5): Test the above deployed web application

Admin console → Deployments → launch VtApp.war file → select personal.html file related URL

Step(6): Undeploy the web application in web logic

Admin console → Deployments →  VtApp → delete

\* To perform hard deployment in Adv.javaBatchDomain of weblogic

server copy webapplications directory on its war file to oracle weblogic\_(VtApp)

home\user\_projects\domains\autodeploy folder.

\* when war file is used for hard deployment (VtApp.war) use the following request URL to test the application.

http://localhost:7070/VtApp/Personal.html

\* The web application that is deployed in weblogic server through hard deployment process can not be undeployed from admin console but can be undeployment through hard deployment process itself.

\* when web application is hard deployed in web logic server through directory based hard deployment (copying VoterApp folder to autodeploy folder) then use following request URL to test the application.

http://localhost:7070/VoterApp/Personal.html

↓  
Directory name as context path

\* In web logic server when you modify the source code of servlet programs placed in deployed web application, just recompilation is enough to get the effect of modifications and there is no need of performing the reloading of web application.

\* The web application and its web resource programs that are having relative URLs is called as WODA applications.

+ Each server supplied separate implementation classes implementing various interfaces of servlet, JSP APIs but we never specify these implementation class names in our servlet programs to make our servlet programs as WODA programs.

### GlassFish

+ type: Application server s/w

Vendor: sun microsystems

version: 2.x (compatible with jdk 1.6/1.5)

open source s/w  
allows to create domains. default domain is domain1  
jar file that represents all see apis: javaee.jar  
default port no: 8001 (for accessing web applications)  
4848 (for accessing admin console)

\* The GlassFish 2.x software that comes with netBeans 6.7.1 IDE can be used with or without IDE.

Procedure to change the default http protocol service related portno in domain1 server of Glass Fish

Go to GlassFish-home\AppServer\domains\domain1\config\domain.xml file and change port attribute value of first<http-listener> tag.

Procedure to perform console deployment of web application in domain1 server of Glass Fish

Step(1): Preparing war file representing web application like vtApp.war

Step(2): start domain1 server of Glass Fish.

start → Programs → Sun Microsystems → Application Server → start default server

Step(3): open admin console of domain1 server

open browser window → type http://localhost:4848

username :   
password :

Step(4): deploy the web application

Admin console → applications → web applications → deploy →

Type :  [web application (.war)]

location : Browse and select vtApp.war → ok

Step(5): Test the web application

open the browser window → type below URL

http://localhost:5151/vtApp/Personal.htm

war file name as context path

\* The Glassfish server supports only war file based hard deployment and it does not support directory based hard deployment.

### Procedure to perform hard deployment of web application in GlassFish 2.x server.

Step(1): create war file representing the web application (like vtApp.war)

Step(2): start domain server of Glassfish.

start → programs → Sun Microsystems → Application Server → start default server.

Step(3): Deploy the web application.

copy the above vtApp.war file to <GlassFish\_home>\Appserver\Domains\domain1\autodeploy folder.

Step(4): Test the application.

open browser window → type below URL.

http://localhost:5151/vtApp/personal.html

war file name as context path.

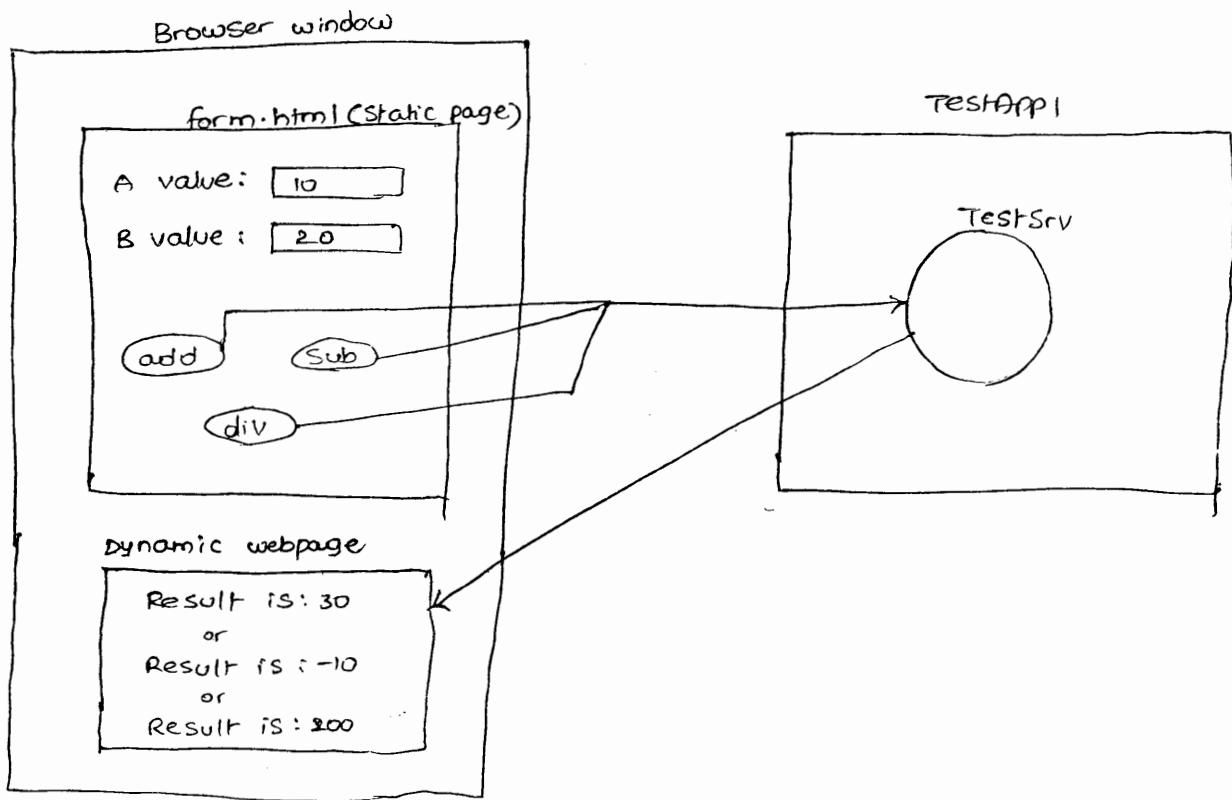
\* When form page submit button is taken with logical name then the caption of the submit button will go to server as request parameter value. otherwise submit button caption never goes to server as request parameter value.

```
<form action="vturl" method="get">
    — — —
    — — —
    <input type="submit" name="s1" value="check">
</form>
```

Here s1=check will go to server as request parameter name and value.

```
<form action="vturl" method="get">
    — — —
    <input type="submit" value="check">
</form>
```

NOTE: Here related to submit button no value will go to server as request parameter value.



To handle the above form page related request processing give same name to all the three submit buttons with different captions and use that caption as criteria value in servlet program to differentiate request processing logic for each submit button.

### form.html

```
<form action="turl" method="get">
    A value: <input type="text" name="t1"> <br>
    B value: <input type="text" name="t2"> <br>
    <input type="submit" name="S1" value="add">
    <input type="submit" name="S2" value="sub">
    <input type="submit" name="S3" value="div">
</form>
```

### TestSrv.java

```
public class TestSrv extends HttpServlet
{
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
```

```

response.setContentType("text/html");
printWriter pw = response.getWriter();
//read form data
int a = Integer.parseInt(request.getParameter("t1"));
int b = Integer.parseInt(request.getParameter("t2"));
//read caption of submit button
String cap = request.getParameter("S1");
if(cap.equals("add"))
{
    pw.println("sum is & result is :" + (a+b));
    pw.println("add btn is clicked");
}
else if(cap.equals("sub"))
{
    pw.println("result is : " + (a-b));
    pw.println("sub btn is clicked");
}
else
{
    pw.println("result is : " + (a/b));
    pw.println("div btn is clicked");
}
//close stream object
pw.close();
}

```

### web.xml

Configure TestSrv servlet program with "/turl" pattern

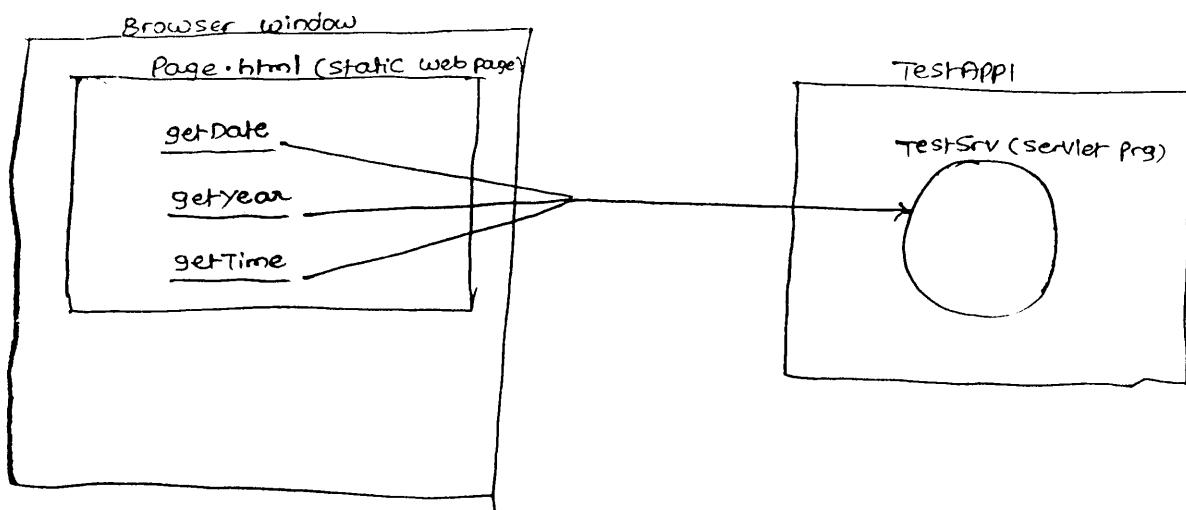


fig: Handling with multiple Hyperlinks

### Page.html

```
<form action="/">  
<a href = "turl? P1=link1" >getDate </a>  
<br><br>           query string  
<a href = "turl ? P1 = link2" >getYear </a>  
<br><br>  
<a href = "turl ? P1 = link3" >getTime </a>
```

NOTE: The query string that is appended to the URL of href attribute sends request parameter to server along with the request.

### TestSrv.java

```
public class TestSrv extends HttpServlet  
{  
    protected void service(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException  
    {  
        //general settings  
        response.setContentType("text/html");  
        PrintWriter pw = response.getWriter();  
  
        //read P1 request parameter value  
        String pval = request.getParameter("P1");  
  
        //write req processing logic based on hyperlink that is clicked  
        Calendar cl = new Calendar.getInstance(); //give sys date  
        if (pval.equals("link1"))  
        {  
            pw.println("link1 is clicked");  
            pw.println("current day is :" + cl.get(Calendar.DAY_OF_MONTH));  
        }  
        else if (pval.equals("link2"))  
        {  
            pw.println("link2 is clicked");  
            pw.println("current year is :" + cl.get(Calendar.YEAR));  
        }  
        else  
        {  
            pw.println("link3 is clicked");  
            pw.println("current time is :" + cl.get(Calendar.HOUR) + ":" +  
                      cl.get(Calendar.MINUTE));  
        }  
    }  
}
```

```
//close stream object  
pw.close();
```

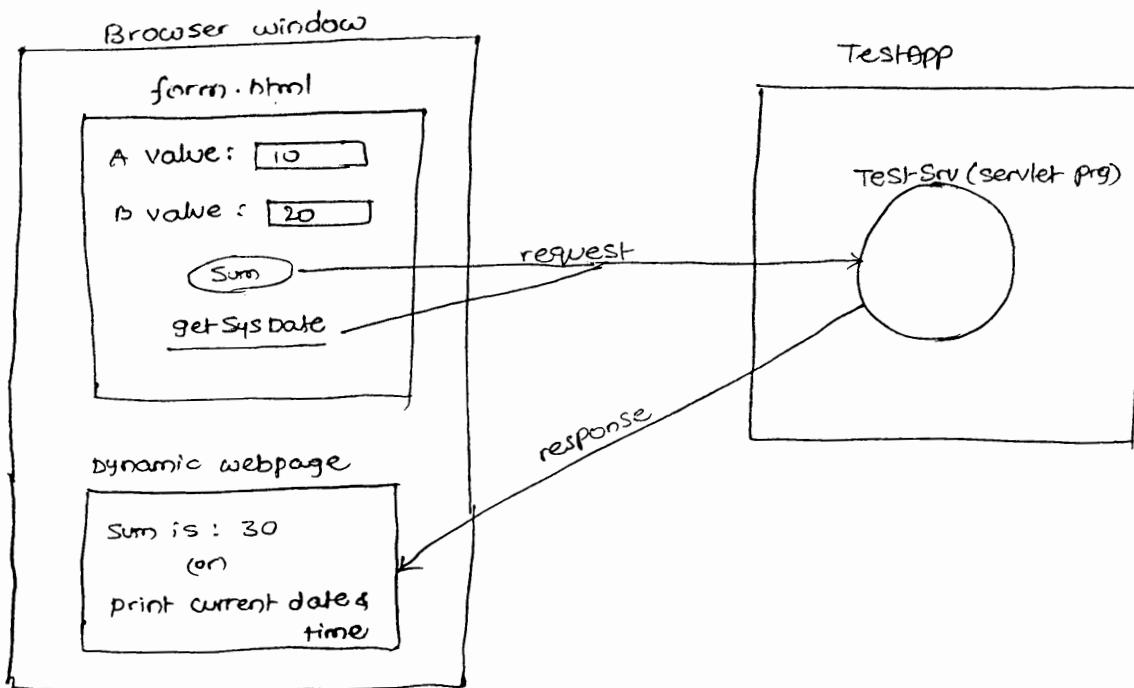
3

3

### web.xml

configure TestSrv servlet program with "/ur" url pattern.

### scenario (3) (Handling both Hyperlink, submit buttons based form page generated request)



\* Give same name for Submit button and hyperlink related additional request parameter having two different values. use that value as criteria value in servlet program to differentiate the logics for submit button , hyperlinks.

\* `Calendar cl=Calendar.getInstance();`

This `getInstance()` is static method of `Calendar` class returning one sub class object of `Calendar` class as return value. `Calendar` class is an abstract class.

\* In the above statement `cl` is not the object of `Calendar` class it is one subclass object of `Calendar` class.

### form.html

```
<form action = "turl" method="get">
    A value <input type = "text" name = "t1"> <br>
    B value <input type = "text" name = "t2"> <br>
    <input type = "submit" name = "s1" value = "add">
</form>
<br> <br>
<a href = "turl? s1= link1" > getSysDate </a>
```

### TestSrv.java

```
public class TestSrv extends HttpServlet
{
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        //general settings
        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        //read s1 req parameter value
        String pval = request.getParameter("s1");
        //write req. processing logic based on hyperlink or submit
        //button on that is clicked
        if (pval.equals("add")) //if add btn is clicked
        {
            //read form data
            int a = Integer.parseInt(request.getParameter("t1"));
            int b = Integer.parseInt(request.getParameter("t2"));
            pw.println("sum is :" + (a+b));
        }
        else if (pval.equals("link1")) //if hyperlink is clicked
        {
            Date d = new Date();
            pw.println("Date is :" + d);
        }
        //close Stream object
        pw.close();
    }
}
```

### web.xml

\* same as previous application.

\* A servlet program is identified through its URL pattern according to servlet specification given by sun microsystems there is a possibility of giving three types of URL patterns.

(1) Exact match

(2) Directory match

(3) Extension match.

### Exactmatch

\* URL pattern begins with "/" symbol and should not contain \* symbol.  
\* multiple words can be there in the URL pattern separated with "/" symbol.

### Ex: In web.xml

<url-pattern> /test1 </url-pattern>

request URLs from browser window to request the above servlet program

http://localhost:2020 /DateApp/test1 (valid)

http://localhost:2020 /DateApp/xyz/test1 (invalid)

/DateApp/test1/abc (invalid)

/Date APP/abc 2 (invalid)

### other examples

ex(1) <url-pattern> /test1/abc </url-pattern>

ex(2) <url-pattern> /abc/test1/xyz </url-pattern>

ex(3) <url-pattern> /test1.cpp </url-pattern>

\* the URL pattern provided for servlet programs can hide the servlet technology and the class names of servlet programs from the end users of the websites. This gives little bit prevention for website from hackers and jackers.

### Directory Match

\* The URL pattern must begin with "/" symbol and must end with "\*" symbol and can also have multiple words separated with "/" symbol.

Ex(1) : /test1/xyz1/\*

request URLs from browser window to request the above servlet program

http://localhost:2020/DateApp/test1/xyz1/abc (valid)

/DateApp/xyz1/test1/abc (invalid)

/DateApp/test1/xyz1/abc.c (valid)

/DateAPP/x/y/test.do (invalid)

/DateApp/test1/xyz1 (valid)

/DateAPP/test1 (invalid)

### other example directory match URL pattern

Ex(1): <url-pattern>/x/y/\*</url-pattern>

Ex(2): <url-pattern>/abc/xyz/\*</url-pattern>

Ex(3): <url-pattern>/abc/\*</url-pattern>

### Extension Match URL pattern

\* URL pattern must begin with \* symbol and must end with extension word/letter

Ex. <url-pattern>\*.do</url-pattern>

↓  
extension word

request URLs from browser window to request the above servlet program

http://localhost:2020/DateApp/abc.do (valid)

/DateApp/abc/xyz/abc1.do (valid)

/DateAPP/.do (valid)

/DateAPP/abc.do/xyz (invalid)

/DateApp/abc.do/xyz.do (valid)

/DateAPP/abc.xyz (invalid)

## other example extension match URL patterns

Ex(1): <url-pattern> \*.abc </url-pattern>

Ex(2): <url-pattern> \*.a </url-pattern>

\* we can not prepare URL pattern of a servlet program by mixing up multiple styles because all servers are designed to just recognize only the above 3 styles of URL patterns.

<url-pattern> /xyz/abc/\*.\*bc </url-pattern>



Invalid URL pattern formation

\* If two servlets are configured with two different styles of URL patterns like exact match and extension match if matching same request URL given by browser window then server gives ~~importance~~ priority to exactmatch url pattern based servlet.

## Example scenario

### In web.xml

DateSrv → with /test.do url pattern (exactmatch)

DateSrv1 → with /\*.do url pattern (extensionmatch)

request from browser window

http://localhost:2020/DateApp/test.do



This test.do matches both DateSrv, DateSrv1 servlet program but priority will be given to DateSrv program with URL pattern exactmatch.

MyEclipse = Eclipse + Built - Plugins to work with advanced technologies

\* A plugin is patch software or software application that can enhance the functionalities of existing s/w and s/w applications.

\* In Java environment plugins come as jar files. In IDE softwares plug ins provides wizards to develop s/w technologies based applications

what is the difference between Eclipse IDE and MyEclipse IDE?

### Eclipse

- (1) open source
- (2) provides environment to develop JSE module applications.
- (3) Does not provide built-in plugin to work with advanced technologies we must supply them manually.
- (4) suitable for small scale companies.

### MyEclipse

- (1) Commercial
- (2) Provides environment to develop JSE, JEE and frame software based applications.
- (3) It provides built-in plugin to work with advanced technologies and also allows to add more plugins explicitly.
- (4) Suitable for medium scale and large scale companies.

### Basic information of myEclipse:

Type : IDE software for java environment.

Version : 8.x (compatible with jdk 1.6)

Vendor : Eclipse organization.

#### Commercial software

Gives tomcat as built-in server. But also allows the programmer to configure other external servers.

To download software : [www.myeclipseide.com](http://www.myeclipseide.com)

for documents : [www.myeclipseide.com](http://www.myeclipseide.com)

### MyEclipse 8.x cheat code:

Subscriber : gocto.cn

Subscription code : TLR8ZC-855444-6666535739876142

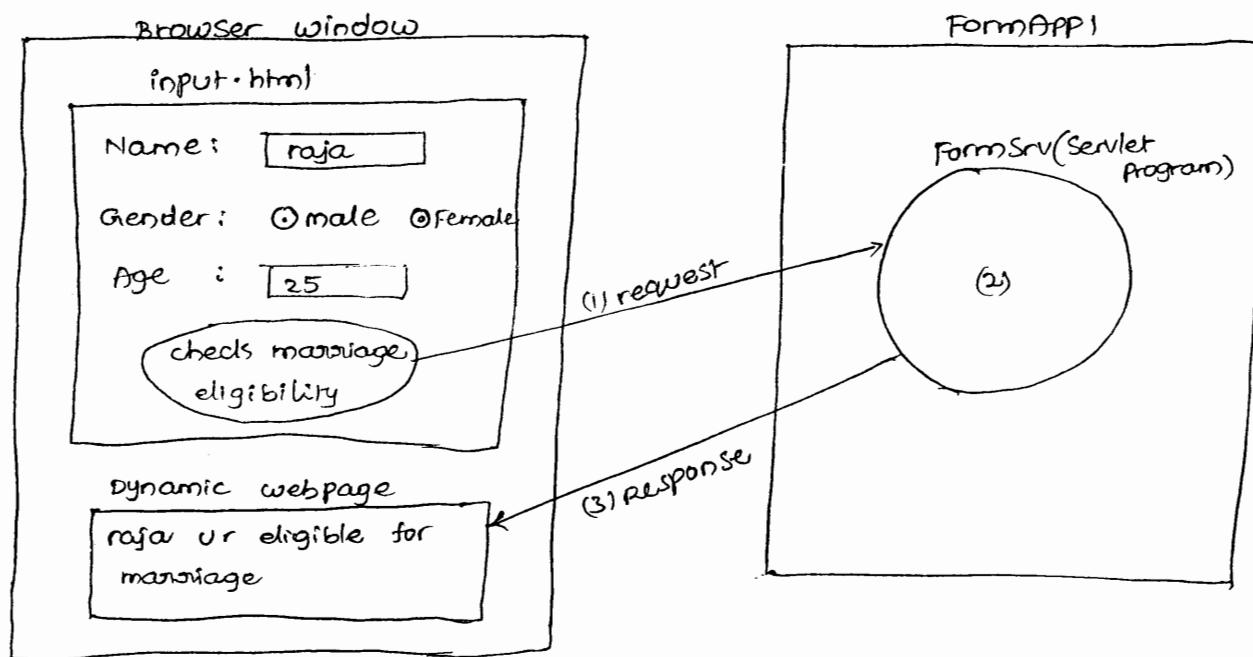
\* Eclipse Gelelio, EG Eclipse are alternate IDEs for myEclipse which are given based on Eclipse IDE.

### Procedure to develop below application by using myEclipse IDE

Step 1: Launch myEclipse IDE by choosing one work space folder to save the projects.

work space : E:\myEclipse 8.x

## Example application



Step (2): Submit the cheat code

MyEclipse menu → Subscription information →

Subscriber: \_\_\_\_\_

Subscription code: \_\_\_\_\_

→ Finish

Step (3): create web project in myEclipse IDE.

File menu → New → web project →

project name: FormApp1

Context root URL: /FormApp1

→ Finish

Step (4): Add form.html to web root folder of the project

Expand project → Right click on web root folder → new → html → filename

form.html

```
<form method = "get" action = "furl" name = "f1">
```

```
    Name: <input type = "text" name = "pname" /> <br/>
```

```
    Gender: <input type = "radio" name = "g1" value = "m" checked /> male
```

```
                <input type = "radio" name = "g1" value = "F" /> female <br />
```

Age: <input type="text" name="page"/> <br/>

<input type="submit" value="checked marriage eligibility"/>  
</form>

\* Add servlet Program to the scr folder of the project.

Right click on scr folder → new → servlet

Name: formSrv → select doGet, doPost methods → next →

Servlet/JSP mapping URL: / → Finish

Step(5): In the generated FormSrv servlet keep the following logic and call that method.

In FormSrv.java

```
public void doGet(--) throws SE, IOE
{
    //general settings
    response.setContentType("text/html");
    PrintWriter pw = response.getWriter();
    //read form data
    String name = request.getParameter("pname");
    int age = Integer.parseInt(request.getParameter("page"));
    String gen = request.getParameter("gen");
    if (gen.equals("M"))
    {
        if (age >= 21)
        {
            pw.println("Mr " + name + " u r eligible to marriage");
        }
        else
        {
            pw.println("Mr " + name + " u r not eligible to marriage");
        }
    }
    else if (gen.equals("F"))
    {
        if (age >= 18)
        {
            pw.println("Miss " + name + " u r eligible to marriage");
        }
    }
}
```

```
else
{
    pw.println("miss" + name + " u r not eligible to marriage")
}
} // else if.

} // doGet(-,-)

public void doPost(-,-) throws SE, IOE
{
    doGet(-,-);
}
```

Step(6): Configure Tomcat 6.x server with myEclipse IDE

window menu → Preferences → myeclipse → servers → Tomcat →  
Tomcat 6.x → Tomcat → Tomcat-home directory : D:\Tomcat 6.0  
 Enable  
 Disable  
→ apply → ok.

Step(7): Start the tomcat Server from myEclipse IDE

Go to servers icon in the tool bar → Tomcat 6.x → start.

Step(8): Deploy the above project in Tomcat server.

Go to deploy icon of the tool bar → project FormApp →

add server Tomcat 6.x  → Finish → Ok

Step(9): Test the application.

open browser window → type this URL

http://localhost:2020/FormApp/form.html ↴

Procedure to Configure Adv.javaBatchDomain server of weblogic 10.3 with

MyEclipse IDE :

window menu → preferences → myEclipse → servers → web logic →  
weblogic 10.3 →  enable →

BEA Home directory : c:\oracle\middleware

Administration username: jawaboss

Administration password: jawaboss1

Execution domain root : e:\middleware\user\_Projects\domains\

Adv.javaBatch Domain.

→ Apply → ok.

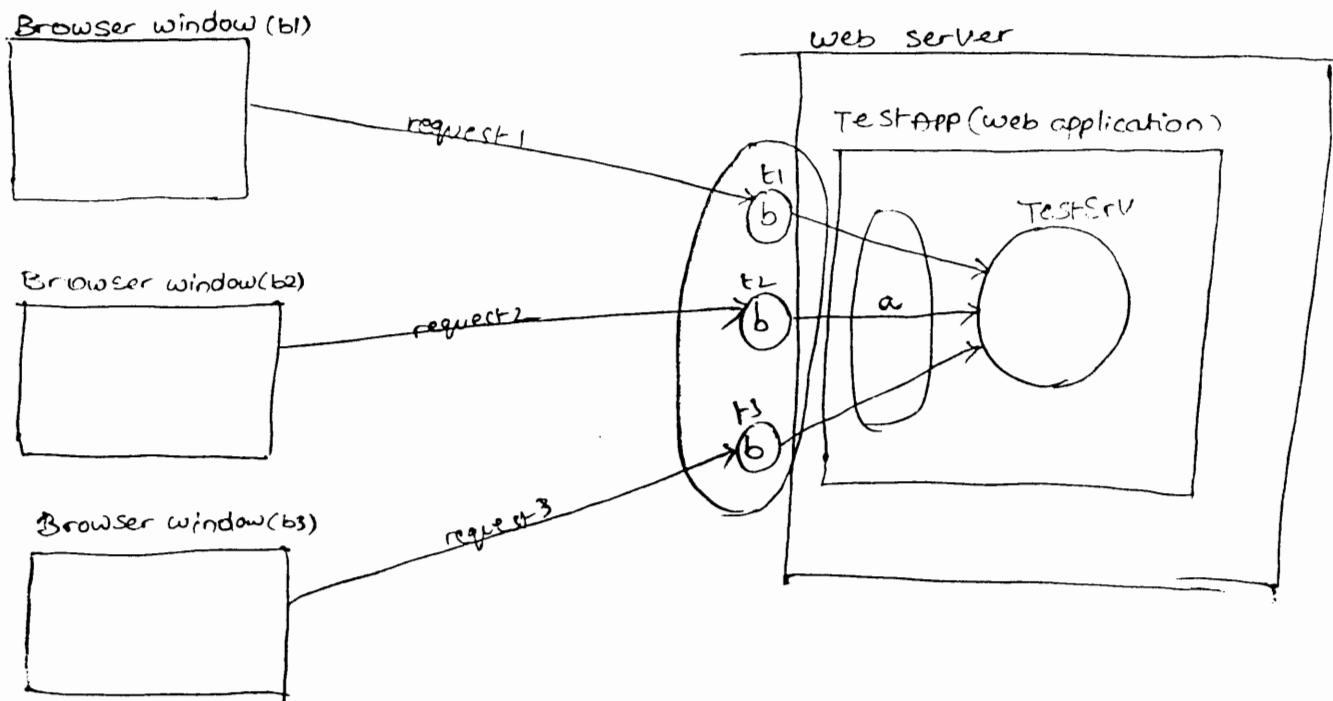
Procedure to configure the domain server of "GlassFish 2.x" with myEclipse IDE:

Window menu → preferences → myEclipse → Servers → GlassFish → GlassFish 2.x →  Enable → Home-directory : D:\Sun\AppServer → Apply → ok.

\* If multiple threads are acting on single variable or object simultaneously or concurrently then we say object/variable is not thread safe. Because, there is a chance of data corruption in that situation.

\* To make the above said variable/object as thread safe use synchronization concepts.

\* Instance variables of our servlet program are not thread safe by default whereas the local variables declared in the service() method of servlet program are thread safe by default.



```

public class TestServlet extends HttpServlet {
    int a;
    public void service( HttpServletRequest request, HttpServletResponse response )
        throws ServletException, IOException {
        int b;
    }
}

```

t<sub>1</sub>, t<sub>2</sub>, t<sub>3</sub> are threads representing requests

\* In the above diagram the multiple threads representing multiple requests of servlet program will act on single copy of instance variable "a" and every thread gets its own copy of local variable "b" so, we can say variable "a" is not thread safe and variable "b" is thread safe.

\* By default our servlet class object and its instance variables are not thread safe. To make them as thread safe, use synchronization concept.

\* To store results, inputs of servlets program in a permanent place like database software or to gather input values of servlet program from permanent storage unit we need to make servlet programs interacting with database software where by placing JDBC code in servlets program.

\* Every JDBC code contains three important operations.

(1) Create JDBC Connection object

(2) Use JDBC Connection object to create other JDBC objects and to develop persistence logic

(3) Close JDBC Connection object.

\* There are three approaches to place JDBC code in our servlet program:

#### Approach (1)

(a) Create JDBC Connection object in init()

(b) Use JDBC Connection object and other object in service(-,-) / doXXX(-,-) methods.

(c) close JDBC connection object and other objects in destroy() method.

\* In approach (1) the JDBC connection object must be declared as instance variable so it is not thread safe (it is disadvantage). programmer must use synchronisation concepts to make the connection object as thread safe.

\* In approach (1) all the requests given to servlet program will use single JDBC connection to interact with database software. This is improve the performance. (It is advantage).

### Approach (2):

(a) create JDBC connection object in service(-,-) / doXXX(-,-) methods

(b) use JDBC connection object and other objects in service(-,-) / doXXX(-,-) methods

(c) close JDBC connection object and other objects in service(-,-) / destroy() method.

\* Here JDBC connection object is local variable to service(-,-) method so it is thread safe object (it is advantage).

\* Each request given to servlet program will establish one new connection with database software. This kills the performance (It is disadvantage).

### Approach (3):

(a) Get JDBC connection object from server managed JDBC connection pool being from service(-,-) method / doXXX(-,-) method.

(b) use JDBC connection object and other JDBC objects in service(-,-) / doXXX(-,-) methods.

(c) Return JDBC connection object back to JDBC connection pool from service(-,-) / doXXX(-,-) methods.

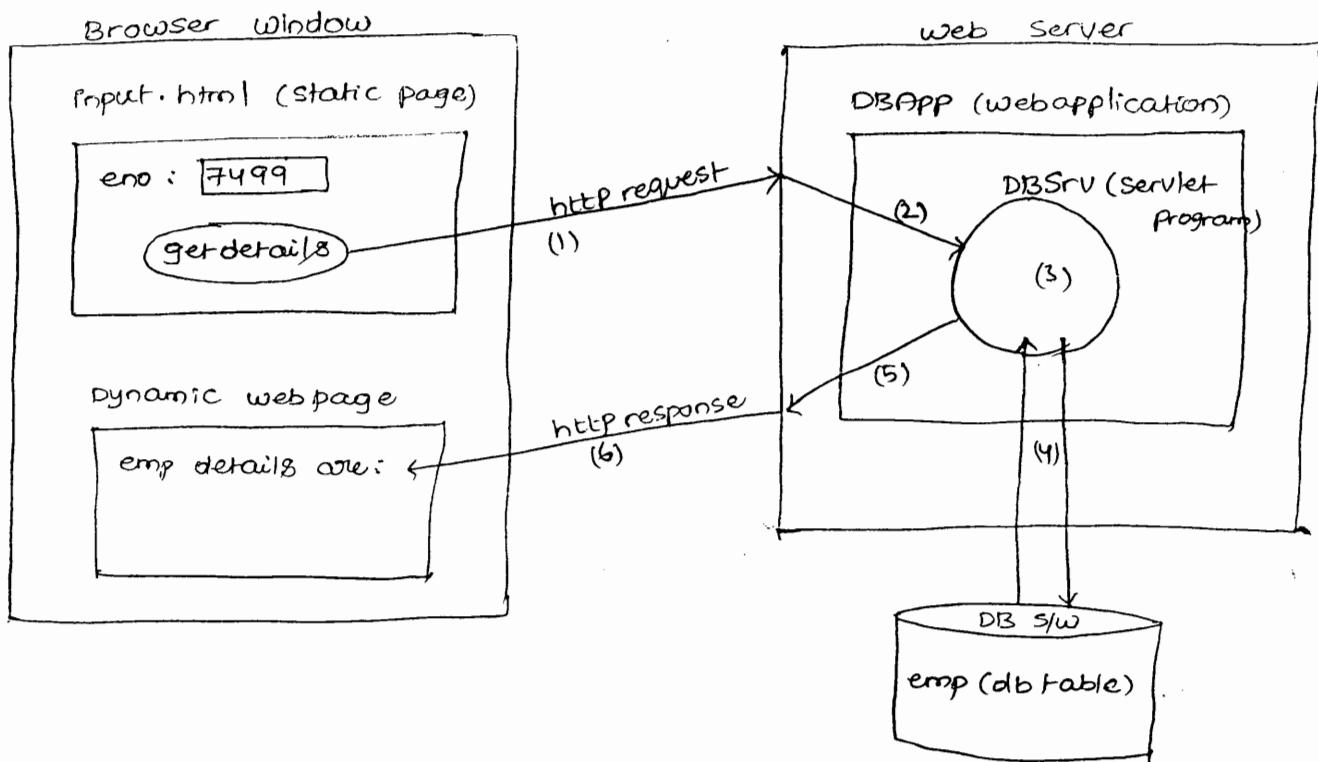
\* Here Connection object is local variable of service(-,-) method. so it acts as thread safe object. (It is advantage).

\* By using minimum no. of JDBC connection objects of connection pool we can make more servlet programs and all requests coming to

○ servlet programs interacting with database software. This gives better performance (it is also advantage).

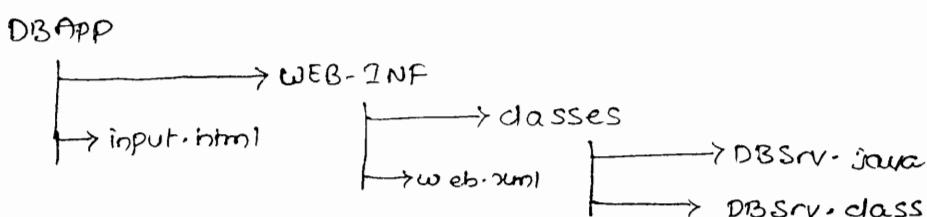
\* Approach (3) is the best for servlet to database software communication.

### Example application on servlet to database software communication

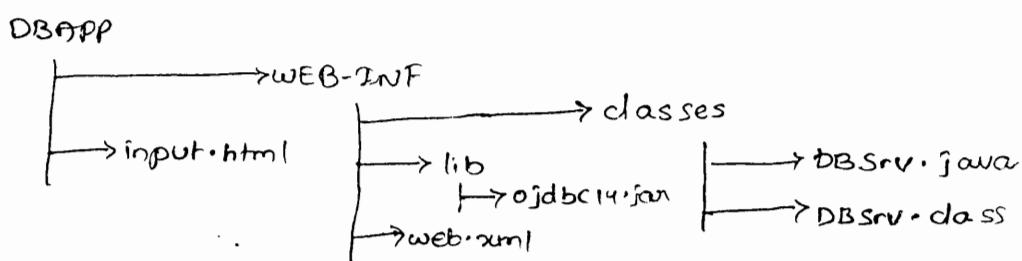


### Deployment structure

when DBSrv program uses type 1 JDBC driver:



when DBSrv program uses type 4 (oracle thin) JDBC driver



- \* when stand-alone Java application uses 3rd party API (other than JDK APIs) then the third party API related jar files must be placed in class path to make Java Compiler and JRE to recognize and use third party API.
- \* If Java web application uses third party API in its web server resource programs (like servlet, JSP Programs) then third party API related jar file should be added to class path and should also be added to WEB-INF\lib folder of web application. Here jar files added to class path will be used by javac to recognize third party API during the compilation of servlet programs. Similarly jar files added to WEB-INF\lib folder will be used by servlet Container to recognize and use third party API during the execution of servlet programs.  
Ex: If servlet program uses oracle thin driver then keep ojdbc14.jar file in class path and WEB-INF\lib folder of web application as shown above.
- \* the jar files added to class-path are not visible in IDE softwares and in container softwares of servers.
- \* Stand alone Java application will be compiled and executed from Command prompt whereas servlet program compilation takes place in servlet Container of Servlet Server/Application Server.
- \* If above web application is created in IDE software then the third party API related jar file must be placed in the libraries folder of the project.

Source code of above diagram based application

input.html

```
<form action="dburl" method="get">  
Employee no: <input type="text" name="teno"> <br>  
<input type="submit" value="getEmpDetails">  
</form>
```

## DBSrv.java (to interact with DB s/w using type 4 driver and approach 1)

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
public class DBSrv extends HttpServlet
{
    Connection con=null;
    PreparedStatement ps=null;
    public void init()
    {
        try
        {
            //create jdbc connection object
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:
                                            satya", "scott", "tiger");

            ps=con.prepareStatement("select ename, job, sal from Emp
                                  where empno = ?");
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
        ServletException, IOException
    {
        try
        {
            //read form data
            int no=Integer.parseInt(req.getParameter("eno"));
            //write business logic
            //set parameter values to SQL query
            ps.setInt(1,no);
            //execute the query
            ResultSet rs=ps.executeQuery();
            //process the resultSet object
            String name=null, desg=null, bsal=null;
```

```

if (rs.next())
{
    name = rs.getString(1);
    desg = rs.getString(2);
    bsal = rs.getString(3);
}
//display emp details as web page Content
PrintWriter pw = res.getWriter();
res.setContentType("text/html");
pw.println("Emp details: <br>");
pw.println("<br> Emp Name: " + name);
pw.println("<br> Emp salary: " + bsal);
pw.println("<br> Emp desg : " + desg);
//close stream objects
pw.close();
rs.close();
}//try
catch (Exception e)
{
    e.printStackTrace();
}
// doGet(-,-)
public void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
{
    doGet(req, res);
}
//doPost(-,-)
public void destroy()
{
    try {
        if (ps != null)
            ps.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    try {
        if (con != null)
            con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
// class

```

### web.xml

configure DBSRV servlet program  
with "/dburl" as url-pattern.

### Request URL to test application

http://localhost:2020/DBApp/input.html

\*If multiple web applications of a server are utilizing same third party API (like oracle thin driver) then instead of keeping the 3rd party API related jar file (like `ojdbc14.jar`) in `WEB-INF\lib` folder of every application it is recommended to place only once in common library folder of server software instantiation.

#### Common library folder in Tomcat 6.0

`<Tomcat-home>\lib` folder

#### Common library folder in Tomcat 5.0/5.5

`<Tomcat-home>\common\lib` folder

`<Tomcat-home>\server\lib` folder

#### Common library folder in weblogic (Any Version)

`<BEA-home>\wlserver-<version>\common\lib` folder

#### Common library folder in GlassFish 2.x (Any Version)

`<GlassFish-home>\AppServer\lib` folder

\*when web resource program of web application uses the third party API class or interface then the servlet container looks for third party API class or interface in the following places and in the following order

- In `WEB-INF\classes` folder itself (if not available then (ii))
- In the jar files added to `WEB-INF\lib` folder (if not available then (iii))
- In the jar files added to common library folder of underlying server. (like `Tomcat-home\lib` folder) (if here also not coming then class not found exception is occurred).

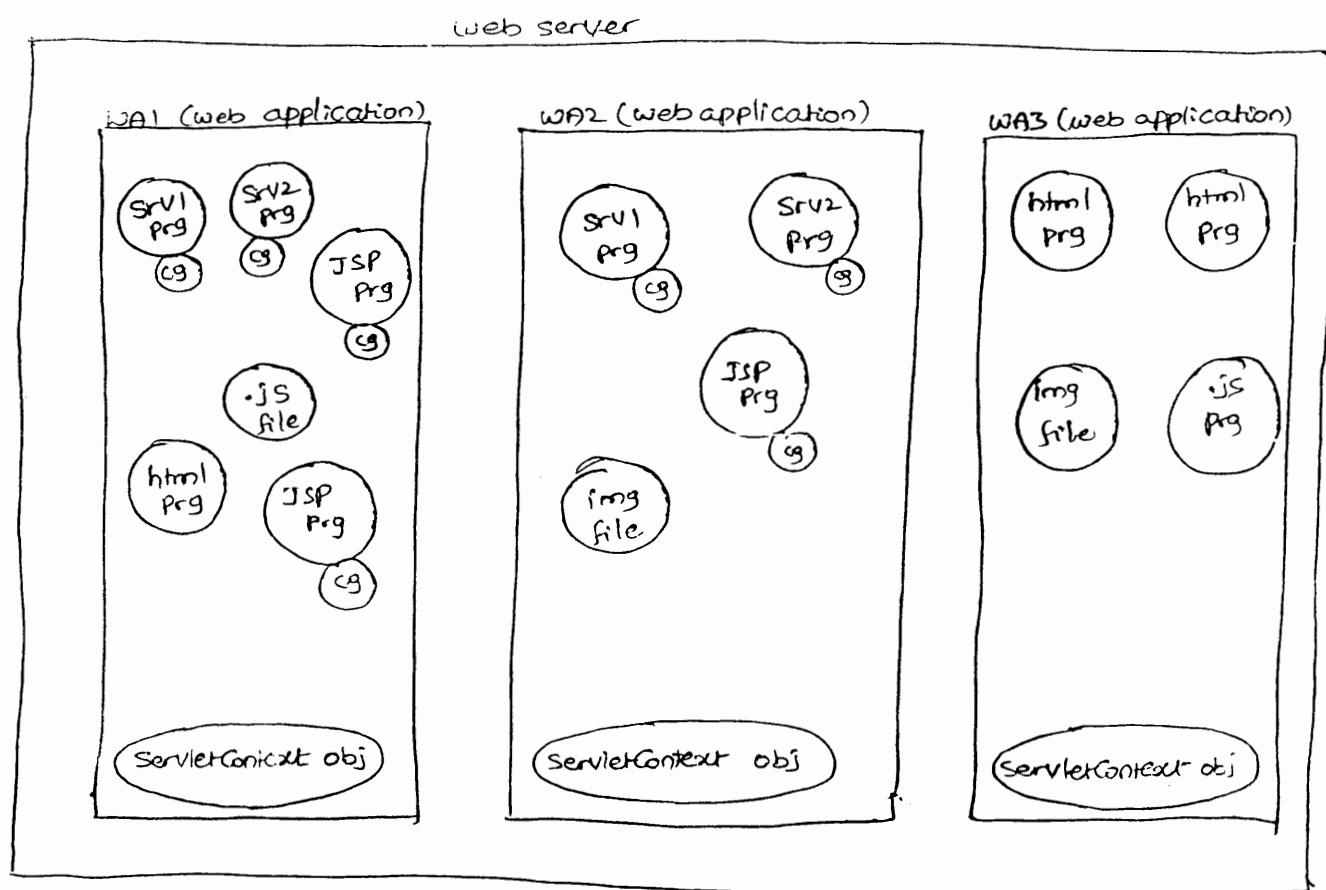
\*Every server internally uses one jdk software so then servlet program uses type-1 jdbc driver to interact with database software then there is no need of adding any jar files in `WEB-INF\lib` folder and classpath.

what is the difference between `servletContext` object and `ServletConfig` object?

(A) Servlet Config object (cg):

\*It is one for our servlet program / JSP program.

- \* ServletConfig object means it is the object of servlet container supplied java class (implementing) implementing javax.servlet.ServletConfig interface.
- \* This object is useful to pass additional data to servlet and to read additional details from Servlet.
- \* This object is useful to read Servlet init parameter values from web.xml file of the web application.
- \* Servlet Container creates the ServletConfig object in the instantiation and initialization process of our servlet class object.
- \* Servlet Container destroys our servlet class object in the destruction process of our servlet object.



cg - ServletConfig object

.js file - javascript code

img file - image file

Srv Prg - servlet program

## Servlet Context object

- \* It is one per web application. So it is called as the global memory of web application.
- \* servlet context object means it is the object of a java class (Container supplier) implementing javax.servlet.ServletContext interface.
- \* servlet container creates this object either during deployment of the web application or during server start up.
- \* Servlet container destroys this object automatically when web application is undeployed or reloaded or stopped or when server is stopped/re-started.
- \* Using this object we can read global init parameters or context parameters from the web.xml file of the web application.
- \* Using this object we can know the details of underlying server like server name, version and the servlet-api version supported by the server.
- \* Using this object we can get context path of the current web application and absolute path of the any web resource program in web application.
- \* The data kept in servletContext object is visible and accessible in all servlet, JSP programs of web application.

In one web server 10 web applications are deployed. In that six web applications are there in running mode and four web applications are there in stopped mode. Can you tell me how many servletContext objects are currently available in that web application? server?

(A) Six

There is a web application with 10 servlet programs. In that three servlet programs are already requested by clients and other 3 servlet programs are enabled with load-on-startup. Can you tell me how many servletConfig objects are currently available in that web application?

(A)  $3+3 = 6$

NOTE: Servlet Container creates ServletConfig object for Servlet program only when the class of servlet program is instantiated (object creation).

\* Our servlet class object, request object, response object, ServletConfig object, ServletContext object can not be created by programmer manually. The servlet container creates all these objects but programmer can get access to these objects in servlet program.

+ To get access to our servlet class object use "this" keyword -

\* To get access to request, response objects use parameters of service(-) / doXXX (-,-)

\* To get access to ServletConfig obj

approach(1):

```
public class TestSrv extends HttpServlet
{
    ServletConfig cg;
    public void service(ServletConfig cg)
    {
        this.cg = cg;
        // use cg here
    }
    public void service(..) / doXXX(..)
    {
        use cg here
    }
}
```

approach(2):

```
public class TestSrv extends HttpServlet
{
    public void init()
    {
        ServletConfig cg = getServletConfig();
        use cg here
    }
    public void service(..) / doXXX(..) method
    {
        ServletConfig cg = getServletConfig();
    }
}
```

NOTE: approach(2) is good.

NOTE: self class methods and super class public, protected methods can be called in the sub class without object.

\* To get access to ServletContext object

approach(1) :

```
public class TestSrv extends HttpServlet
{
    public void init()
    {
        ServletConfig cg = getServletConfig();
        ServletContext sc = cg.getServletContext();
        // use sc here
    }
    public void service(..) / doxxx(..)
    {
        ServletConfig cg = getServletConfig();
        ServletContext sc = cg.getServletContext();
        // use sc here
    }
}
```

= = =

approach(2) :

```
public class TestSrv extends HttpServlet
{
    public void init()
    {
        ServletContext sc = getServletContext();
        // use sc here
    }
    public void service(..) / doxxx(..)
    {
        ServletContext sc = getServletContext();
        // use sc here
    }
}
```

= =

\* Directly or indirectly we need servletConfig object to get access to ServletContext object.

NOTE: Approach(2) is good.

\* `getServletContext()` method definition of predefined GenericServlet class internally uses `servletConfig` object to give `ServletContext` objects.

\* To make JDBC code of servlet program as flexible code to modify it is recommended to gather the following four details of JDBC code from outside the servlet program. They are

(i) JDBC driver class name

(ii) JDBC URL

(iii) DB Username

(iv) DB password.

\* There are two ways to pass input values to servlet program from outside the servlet program.

(1) request parameters / form data.

→ end user / visitor of web application sends these values through browser window (from client side)

→ since end user is non-technical person this data will be non-technical data like name, age, address of end user.

→ servlet program uses `request` object to read request parameter values.

(2) Servlet init parameters

→ programmer passes this data to servlet program from `web.xml` (server side).

→ since programmer is technical person, this data can be technical data like JDBC driver, URL and etc details.

→ servlet program uses `servletConfig` object to read init parameter values of `web.xml`.

\* To make the JDBC code of servlet program as flexible code to modify it is recommended to gather the above said JDBC details (driver class, URL, username, password) from `web.xml` file as Servlet init parameter values.

\* init parameters are specific to each servlet programs and thus programmer choice name and value.

- \* For revised DB APP application that uses servlet init parameter to make jdbc code of DBSrv servlet as flexible code to modify refer application (4) of the page nos 63-65.
- \* send technical input values to servlet program from web.xml file as init parameter values. Send non-technical input values to servlet program as request parameter values / form data.

### Different ways of reading servlet init parameter values

```
ServletConfig cg = getServletConfig();
```

#### approach(1)

```
String s1 = cg.getInitParameter("driver");
```

here we must know init parameter name to get the value

#### approach(2)

```
Enumeration e = cg.getInitParameterNames();
```

```
while (e.hasMoreElements())
```

```
{
```

```
String name = (String) e.nextElement();
```

```
String val = cg.getInitParameter(name);
```

```
}
```

gives all init parameter names and values.

\* when you configure multiple init parameters in web.xml file having same logical name then the last value will be effected as that init parameter value.

\* Using ServletConfig object we can gather the container created object name of our servlet class.

```
S.O.P("current servlet logical name/ instance name" + cg.getServletName());
```

↓  
gives < servlet-name > tag value from web.xml

\* If multiple servlet/ JSP programs of web application are looking to use same init parameters then instead of placing them in every servlet configuration it is recommended to write them only once in web.xml as context parameters (or) global init parameters and use servletContext

object in all servlet/JSP programs to read these context parameter values.

\* If multiple servlet/JSP programs of a web application are talking with DB software by using same JDBC driver to make their JDBC code as flexible code instead of specifying JDBC driver details as init parameters in every servlet program configuration it is recommended to place JDBC driver details only once in web.xml as context parameters as shown below.

### In web.xml of DBApp web application

```
<web-app>
  <context-param>
    <param-name> driver </param-name>
    <param-value> oracle.jdbc.driver.OracleDriver </param-value>
  </context-param>
  <c-p>
    <p-n> dburl </p-n>
    <p-v> jdbc:oracle:thin:@localhost:1521:ord </p-v>
  </c-p>
  <c-p>
    <p-n> dbuser </p-n>
    <p-v> scott </p-v>
  </c-p>
  <c-p>
    <p-n> dbpwd </p-n>
    <p-v> tiger </p-v>
  </c-p>
  < servlet>
    < servlet-name> db </servlet-name>
    < servlet-class> DBSrv </servlet-class>
  </servlet>
  < servlet-mapping>
    < servlet-name> db </servlet-name>
    < url-pattern> /dburl </url-pattern>
  </servlet-mapping>
```

Context parameters of web application

```
<Servlet>
  <Servlet-name> t </Servlet-name>
  <Servlet-class> TestSrv </Servlet-class>
</Servlet>

<s-m>
  <s-n> t </s-n>
  <u-p> /testurl </u-p>
</s-m>

<b-app>
```

In `init()` method of DBSrv Servlet program belonging to DBApp webapplication

```
public void init()
{
    try
    {
        //read init param values from web.xml

        ServletContext sc = getServletContext();

        String s1 = sc.getInitParameter("driver");
        String s2 = sc.getInitParameter("dburl");
        String s3 = sc.getInitParameter("dbuser");
        String s4 = sc.getInitParameter("dbpwd");

        ==
        ==
    }
    //same as previous application code
}
```

NOTE: init parameters are specific to that servlet/JSP program for which they are configured. Context parameters are visible in all web resource programs (java based) of web application.

Different ways of reading context parameter values

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

approach(1) :

```
String SI = sc.getServletInitParameter("driver");
```

here we must know Context parameter name to get the value.

### approach(2)

```
Enumeration e = sc.getInitParameterNames();
while (e.hasMoreElements())
{
    String name = (String)e.nextElement();
    String val = sc.getInitParameter(name);
}
```

//gives all context parameter names and values.

\* we can use same name <sup>for</sup> both init parameter of Servlet program and for Context parameter. we can access both these parameter values in our servlet program using `ServletConfig`, `ServletContext` objects respectively.

\* To gather misc info about underlaying server use `ServletContext` object as shown below.

```
// gathering misc info using ServletContext obj
pw.println("Server info is :" + sc.getServerInfo()); // gives Apache Tomcat 6.0
pw.println("Servlet api spec version impl by underlaying server:" +
           sc.getMajorVersion() + "." + sc.getMinorVersion()); // gives 2.5
pw.println("Context path of current web application:" + sc.getContextPath()); // gives /DBAPP
pw.println("absolute path of input.html on server:" + sc.getRealPath(
           "/input.html")); // gives Tomcat installation path and input.html
```

- \* Tomcat 5.x → given based on servlet api 2.4 spec
- \* Tomcat 6.x → given based on servlet api 2.5 spec
- \* Tomcat 7.x → given based on servlet api 3.0 spec

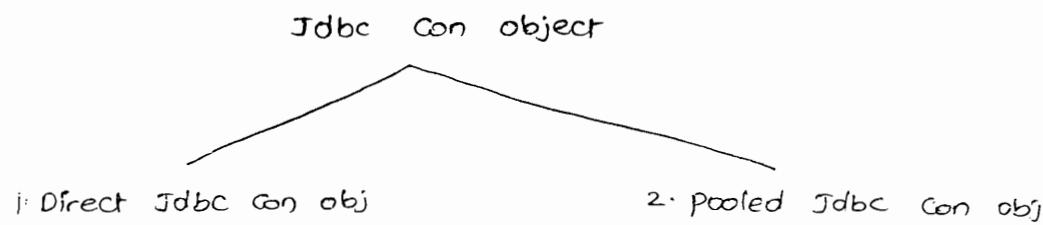
What is the difference between code placed in the constructor and code placed in the `init()` method of our servlet program?

- \* The Servlet Container creates `ServletConfig` object after ~~ex~~ constructor execution and before ~~execut~~ `init()` method execution in the instantiation

and initialization process of servlet program so `ServletConfig` object is not visible in constructor but visible and accessible in the `init()` method.

Since `ServletContext` object can not be accessed without `ServletConfig` object so the `ServletContext` object is not accessible in the constructor but `ServletContext` object is accessible in the `init()` method.

The code placed in the constructor of servlet program can not work with init parameters and context parameters where as the code placed in `init()` method can utilize those parameters so it is always recommended to place the initialization logic only in `init()` method.



#### To create direct Jdbc Con obj

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

```
Connection con = DriverManager.getConnection(url, uname, pwd);
```

\* The Jdbc connection object that is created by programmer manually is called as direct Jdbc Connection object.

\* The Jdbc connection object that is collected from Jdbc connection pool is called as pooled Jdbc connection object.

3 types of Jdbc Con pools.

(1) Driver managed Jdbc Con pool

(2) Third party sw managed Jdbc Connection pool (like Apache DBCP, c3p0, proxool and etc).

(3) Server managed Jdbc Connection pool (in web server, application server environment).

\* Oracle thin driver gives one built-in Jdbc Connection pool in its `ojdbc14.jar` file. That Connection pool is called as `Drivermanaged Connection`.

pool use (1), (2) type JDBC connection pools in stand alone applications.  
use (3) type connection pool in those applications that are deployable  
in the server like web applications, EJB Components & and etc..

- \* In Driver managed, third party s/w managed JDBC connection pool environment we need to use type1/ type2/ type3 JDBC drivers.
- \* In server managed JDBC connection pool environment we need to use type3 with type1/ type2/ type4 JDBC drivers.
- \* Each JDBC Data source object represents one JDBC Connection pool to get each Connection object from Connection pool we need to depend upon this data source object.
- \* JDBC DataSource object means it is the object of a Java class that implements javax.sql.DataSource interface.
- \* All JDBC connection objects of connection pool represents connectivity with same database software. JDBC Connection pool for oracle means all JDBC connection objects in that Connection pool represents connectivity with oracle database software.
- \* The advantage of JDBC connection pool is by using minimum no. of JDBC connection objects we can make more clients talking with DB software.

#### jndi registry s/w

nickname (r)	sathyaw	student obj ref
alias name (r)	apple	customer class obj
jndi name	india	date class obj

- \* The registry s/w can maintain objects / object references having nick names or alias name to provide global visibility to them.
- \* To provide global visibility to JDBC Data source object we place that object reference in registry software.
- \* Java applications use JNDI API to interact with registry s/w.

\* The process of getting object / object reference from registry based on its nickname or alias name is called as look-up operation

\* Jndi api means working with javax.naming and its sub packages.

\* Jndi registry s/w are

Rmi registry

Cos registry

Weblogic registry

Jnp registry and etc...

\* Every application server software comes with one built-in registry s/w.

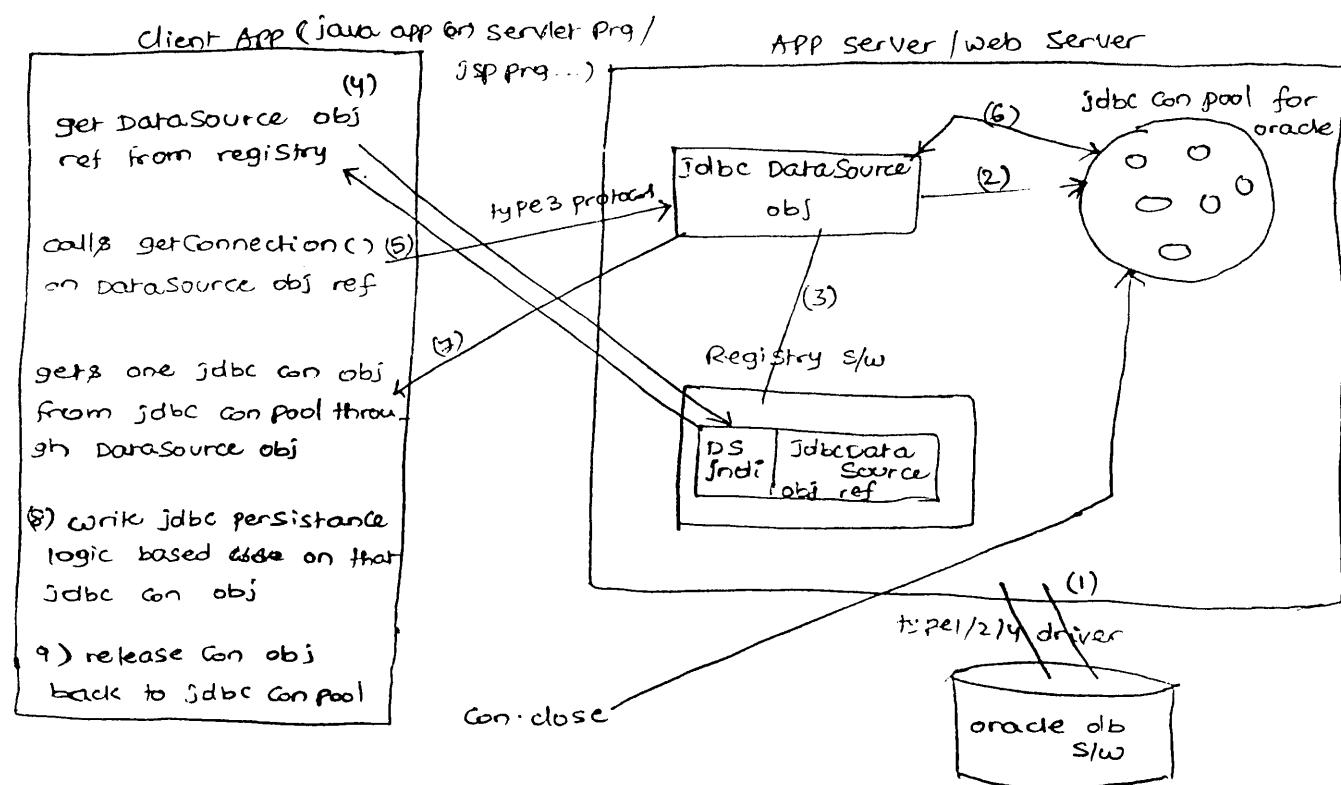
Jboss → Jnp registry

Weblogic → Weblogic registry

Websphere → Cos registry

\* The Connection object of JDBC application represents connectivity between Java application and database software. Similarly the initial context object of JNDI programming represents connectivity between Java application and registry software.

### Understanding the server managed JDBC connection pool environment



\* In server managed JDBC Connection pool environment type3 with type4 JDBC drivers will be utilized. Here type4 will be utilized as a driver to interact with database software and to create connection objects in the connection pool whereas type3 will be used as a protocol by client to get one JDBC connection object from JDBC connection pool.

with respect to diagram

(1) T-L or P-L makes application server to interact with DB SW using type1 or type2 or type4 drivers and creates JDBC connection pool having JDBC connection objects.

(2), (3) → T-L (or) P-L creates dataSource object representing the above JDBC connection pool and also keeps that dataSource object in registry software having nickname for global visibility.

(4) → Client application gets DataSource object reference from registry software through look-up operation.

(5), (6), (7) → Client application calls getConnection() on &DataSource object reference this call uses type3 protocol to get one JDBC connection object from connection pool.

(8) → refer diagram

(9) → refer diagram.

NOTE: The released connection object in JDBC connection pool becomes ready to give service to next request or next client.

Procedure to create JDBC Connection pool for Oracle and JDBC DataSource in GlassFish 2.x Server

Step(1): place jdbc4.jar file in (sun-home\appserver\domains\domain1\lib\ext folder.

Step(2): place mysql related connector's JDBC driver .jar file (mysql-connector-j-3.0.8-Stable-bin.jar file) in GlassFish\home\appserver\domains\domain1\lib\ext folder.

Step(2): start Glassfish server and open its admin console (domain server)

Step(3): create jdbc connection pool for mysql.

Admin console → resources → jdbc → Connection pools → new →

Name : my pool 1

Resource type : javax.sql.DataSource

Database vendor : mysql

→ next → initial and minimum pool size : 82 (initial capacity)

max pool size : 20

pool resize quantity : 2

idle time out : 300

Database name : db1

Password : root

Port : 3306

Port number : 3306

Servername : localhost

URL : jdbc:mysql://localhost:3306/db1

url : jdbc:mysql://localhost:3306/db1

user : root

→ finish → Launch mypool1 (click on mypool1) → Ping

Step(4): create jdbc Datasource representing the above jdbc connection pool

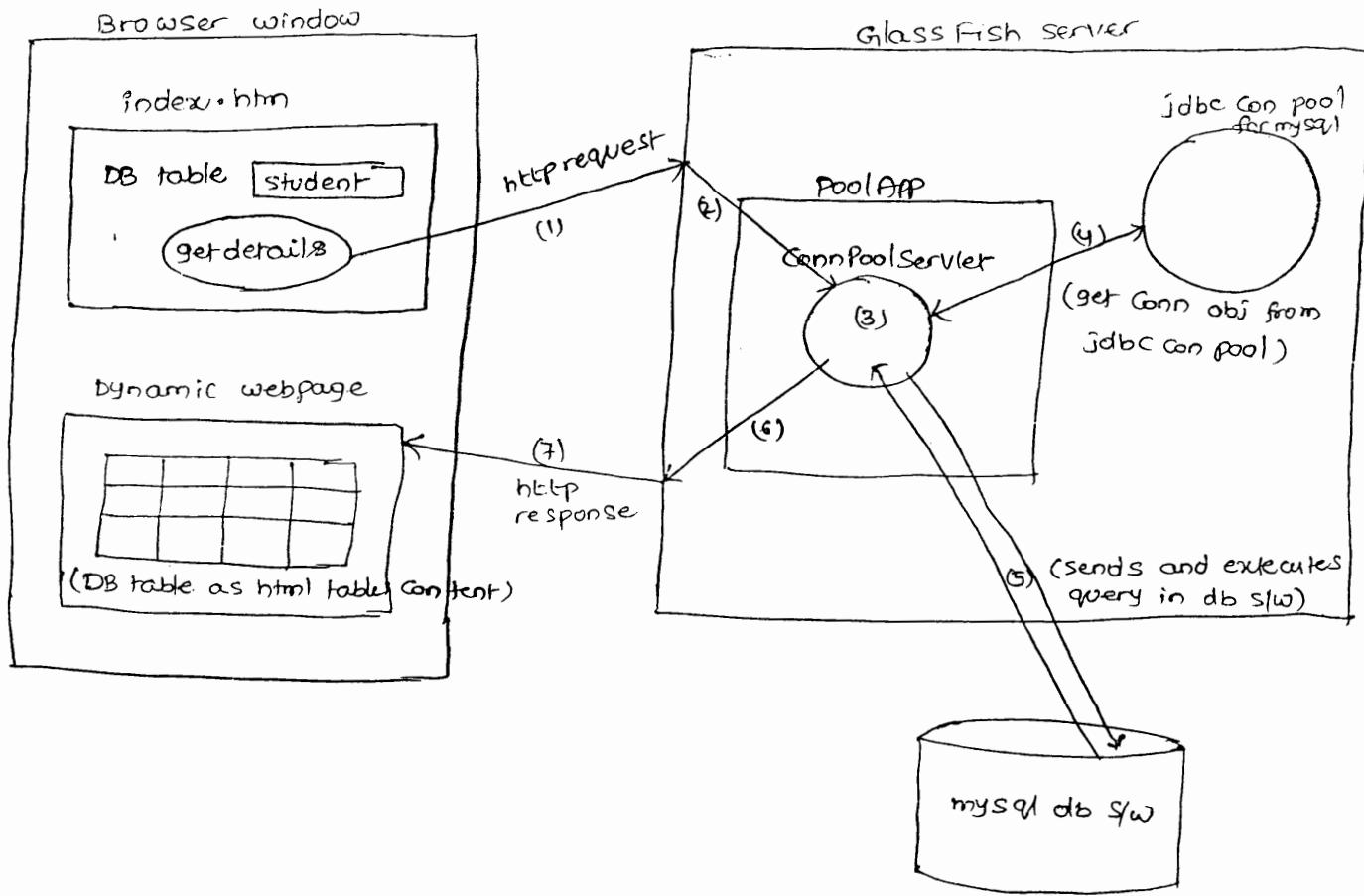
Admin console → resources → jdbc → jdbc resources → new →  
JNDI name: Ds Jndi (any name) (it becomes nickname or alias name  
once datasource object is registered with registry s/w)

pool name : mypool1

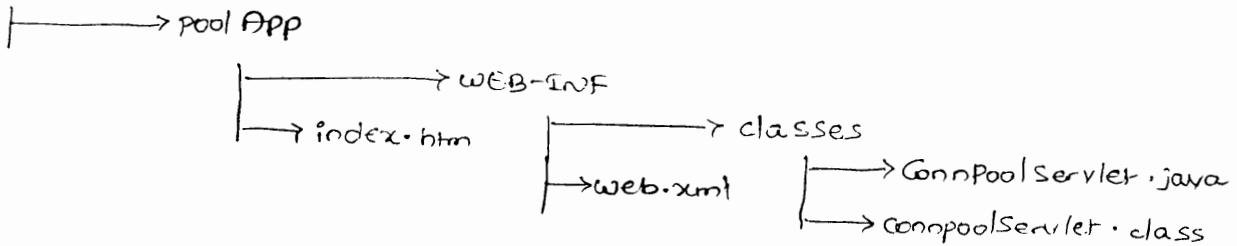
→ ok

NOTE: once ok button is clicked the above created Data Source object will be registered with registry software of GlassFish Server automatically with JNDI name "DsJndi"

Example web application where servlet program uses the Connection object of JDBC Connection pool to interact with DB s/w



E:\Apps



\* Preparing war file

E:\Apps\poolApp>jar cf PoolApp.war .

\* Deploy the above war file in GlassFish 2.x server

Copy .war file to <GlassFish-home>\appserver\domains\domain1\autodeploy folder

Request URL to test the application

(<http://localhost:5151/PoolApp/index.htm>)

\* For above diagram based example application refer application 5 of the page nos 65-67.

\* Always develop web applications by keeping non-technical end-users in mind for that when exception is raised in web resource programs instead of displaying the exception related technical messages on browser window it is recommended to display those messages as non-technical guiding messages for end user. For this we need to write some presentation logic in the catch blocks of servlet programs.

\* Jndi api is part of java JSE module so we need not to add any jar files to classpath while working with Jndi api.

Procedure to create JDBC connection pool for oracle Data source in OracleWebLogic 10.3

Step(1): Start Adv-JavaBatchDomain Server of weblogic and open its admin console  
start → programs → Oracle WebLogic → User-Projects → Adv-JavaBatch Domain → start Admin server for weblogic server domain

Step(2): open browser window the following URL

<http://localhost:7001/console> ↵

user name: Javaboss

password: Javaboss

Step(3): create JDBC data source pointing to the JDBC connection pool for oracle admin console → services → JDBC → DataSources → New →

Name: myds1 (logical name)

JNDI Name : DSJndi (Required for client while performing lookup operation)

database type: Oracle

database driver: "Oracle's Driver (thin) for service connections"

→ next → next

Database name: satya

Host name: localhost

port no: 1521

Database username: scott

Database password: tiger

Conform password: tiger

→ Test Configuration → next → select admin server → finish

NOTE: when finish button is clicked the data source object that represents JDBC connection pool for oracle will be registered with registry software with jndi name "DBJndi".

Step(4): specify the JDBC connection pool additional properties.

Admin console → Services → JDBC → Data sources → myds1 → Connection pool tab

→ initial capacity

maximum capacity

capacity increment  specifies the no. of new JDBC connection objects that should be created at a time if there is a need of creating new JDBC connection objects in the connection pool.

→ save → Advanced → shrink frequency:  seconds →

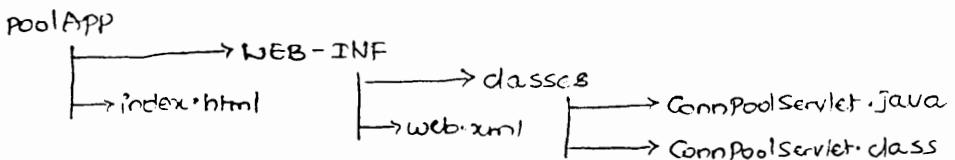
\* For utilizing the above server managed connection pool we can run the application

⑤ of the booklet. For that

Step(i): make sure that the Adv. javaBatchDomain server is in running mode.

Step(ii): make sure that DSJndi is specified as the argument value of ic.lookup(-) method in line no. G12.

Step(iii): Prepare the deployment directory structure or war file representing the web application.



Step(iv): Deploy the above PoolApp application in Adv. javaBatchDomain Server of weblogic.

copy the PoolApp folder to weblogic-home\middleware\user-Projects\Domains\Adv.javaBatchDomain\Auto deploy folder.

Step(v): Test the application

<http://localhost:7001/poolApp/index.html>

### Jboss Server

Type : Application server software

version : 5.x (Compatible with JDK 1.6)

Vendor : Apache / Red Hat

OpenSource software

Default port no: 8080

Allows to create domains and also gives 5 built-in domains. They are

(1) default

(2) web

(3) standard

(4) all

(5) minimal

To download software: download s/w as zip file from [www.apache.org](http://www.apache.org) website.

### Installation process

\* Extract Jboss-5.1.0-GA-Jdk6.zip file to a folder.

\* Jar file that represents all JEE APIs : Jboss-javabe.jar (<jboss-home> client)

### Procedure to change the port no. of "default" domain server of Jboss 5.x

Go to Jboss-Home\server\default\deploy\jbossweb.jar\server.xml file

and modify the port attribute of first <connector> tag.

### TO start jboss server ("default" domain)

use jboss-home\bin\run.bat file.

\* In Jboss server there is no console deployment. But hard deployment

is available (only war file based hard deployment is supported).

### NOTE:

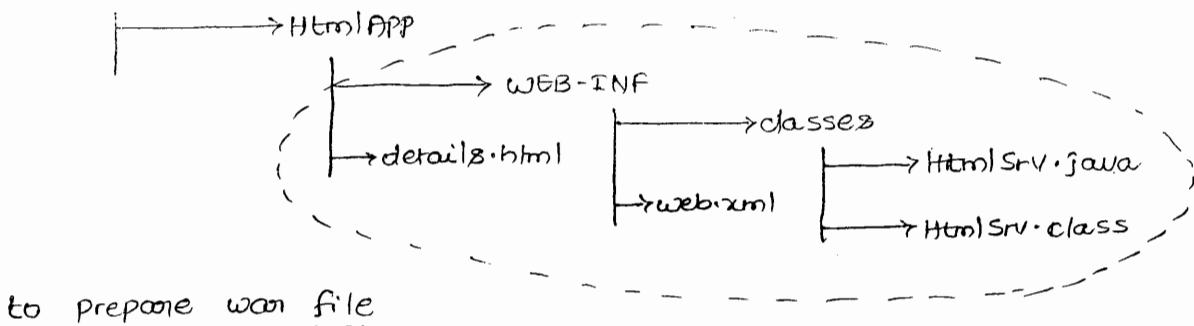
Jboss, GlassFish server directory based hard deployment. They only support .war file based hard deployment.

Procedure to deploy Java web application in 'default' domain of

Jboss 5.x server

Step(1): create .war file on the deployment directory structure of web application.

E:\APPS



E:\APPS\HtmlApp> jar cf HtmlApp.war .

Step(2): start "default" domain Server of jboss.

Step(3): Deploy the web application in jboss server

copy the above "HtmlApp.war" file to jboss-home\server\default\deploy folder.

Step(4): Test the web application

open the browser window and type @this url

http://localhost:8181/HtmlApp/details.html

Common library folder: The common library folder of 'default' domain of Jboss is: jboss-home\server\default\lib folder

Procedure to Configure Jboss with myEclipse IDE:

window menu → preferences → myEclipse → servers →

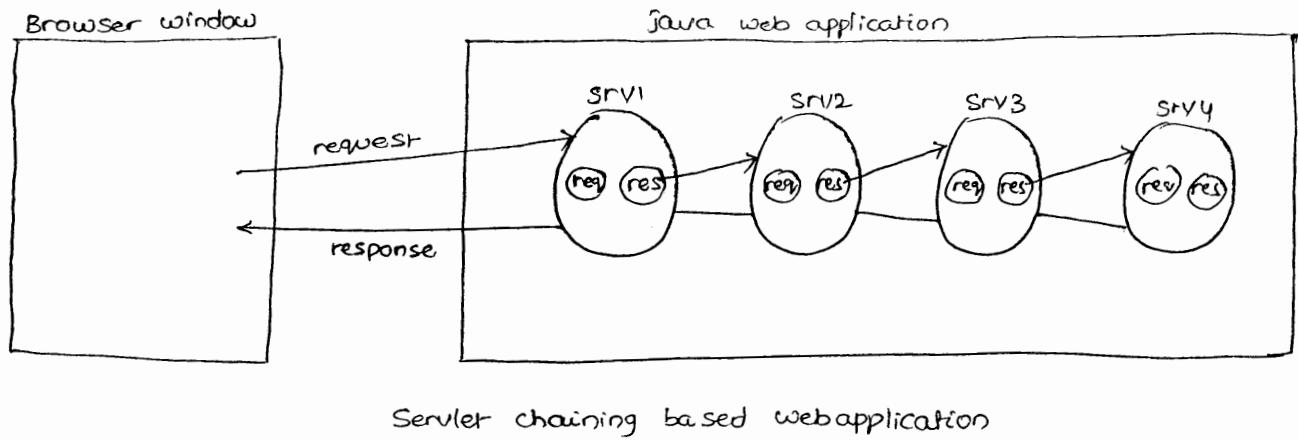
jboss 5.x → Enable Jboss\_home directory: [E:\jboss 5.x soft\jboss-5.1.0.GA]

server name: default (domain name)

→ apply → ok

## 31/10/19 Servlet chaining

- \* Taking request from a browser window and processing that request by using multiple servlets as a chain is called as Servlet chaining.
- In servlet chaining we perform communication between servlet programs to process the request given by a client.

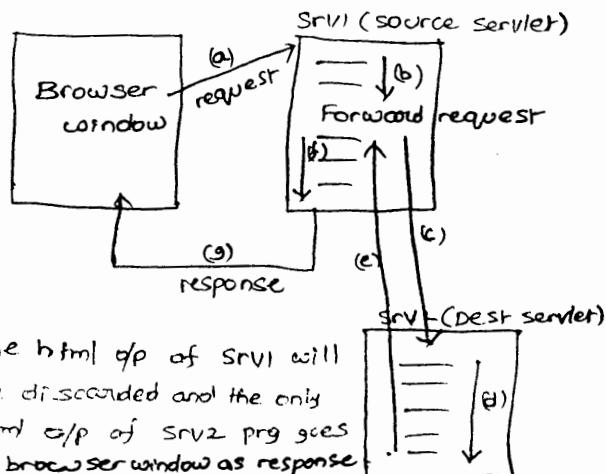


Servlet chaining based webapplication

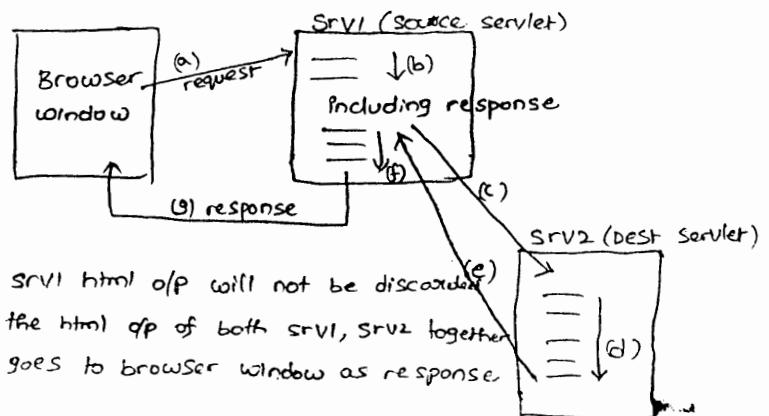
- All servlet programs that participate in a servlet chaining will use same request and response objects because they process the same request that is given by client
- To perform servlet chaining we need RequestDispatcher object. RequestDispatcher object means it is the object of a container supplied java class implementing javax.servlet.RequestDispatcher interface.

### Servlet chaining

#### 1. Forwarding Request mode of servlet chaining



#### 2. including Response mode of servlet chaining.



- \* In any mode of servlet chaining all the servlet programs / web resource programs will use same request and response objects. If srv1, srv2, srv3, srv4 servlet programs are there in forwarding request mode of servlet chaining then the html o/p of srv1, srv2, srv3 programs will be discarded and only the o/p of srv4 servlet program goes to browser window.
  - \* If srv1, srv2, srv3, srv4 servlet programs are there in including response mode of servlet chaining then the html o/p of all servlet programs together goes to browser window as response.
  - \* The source servlet program uses RequestDispatcher object to perform servlet chaining with destination web resource program (like servlet program, JSP program html program and etc...)
- There are 3 approaches to create RequestDispatcher object in source servlet program pointing to destination webresource program.

#### Approach(1) (by using req obj)

Ex: In srv1 source code

```
RequestDispatcher rd = req.getRequestDispatcher (" /s2url ");
rd.forward (req, res);
(or)
rd.include (req, res);
```

↳ URL pattern of destination  
servlet program (srv2)  
↳ optional

Ex(2): In source srv1 Program

```
RequestDispatcher rd = req.getRequestDispatcher (" /abc.html ");
(or)
RequestDispatcher rd = req.getRequestDispatcher (" /abc.jsp ");
rd.forward (req, res);
(or)
rd.include (req, res);
```

#### Approach (2) (by using servletContext obj)

Ex(1): In source srv1 program

```
ServletContext sc = getServletContext ();
RequestDispatcher rd = sc.getRequestDispatcher (" /s2url ");
rd.forward (req, res);
(or)
```

↳ mandatory  
↳ URL pattern of dest svr  
program (srv2)

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

Ex(2): In source svrl program

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

→ mandatory

```
RequestDispatcher rd = sc.getRequestDispatcher("/abc.html");
```

(or)

```
RequestDispatcher rd = sc.getRequestDispatcher("/abc.jsp");
```

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

(or)

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

Approach (3) (by using servletContext obj)

Ex(1): In Src svrl program

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

```
RequestDispatcher rd = sc.getNamedDispatcher("s2");
```

→ logical name of destination  
servlet program (like svrl)  
given in web.xml file

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

(or)

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

Ex(2): In src svrl program

```
RequestDispatcher rd = sc.getNamedDispatcher("j2");
```

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

→ logical name of destination  
JSP Program like abc.jsp

(or)

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

NOTE (1): If source servlet program calls rd.forward(req,res) method then it performs forwarding request mode of servlet chaining with destination web resource program.

NOTE (2): In source servlet program calls rd.include(req,res) method then it performs including response mode of servlet chaining with destination web resource program.

\* servlet Configuration in web.xml is mandatory so it contains logical name and URL pattern.

\* JSP program configuration in web.xml file is optional so it may or may not contain logical name and URL pattern. we can not configure html

program in web.xml file.

what is the difference between getRequestDispatcher() and getNamedDispatcher() methods?

### getRequestDispatcher()

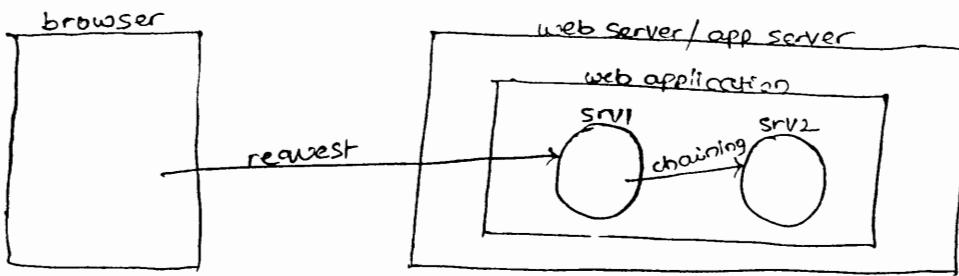
- (1) Invokable on both request, ServletContext object.
- (2) Expects URL pattern of <sup>destination</sup>Servlet program or file names of destination JSP or HTML programs as argument value.
- (3) This method generated RequestDispatcher object can point only the destination servlet, JSP program and HTML program.

### getNamedDispatcher()

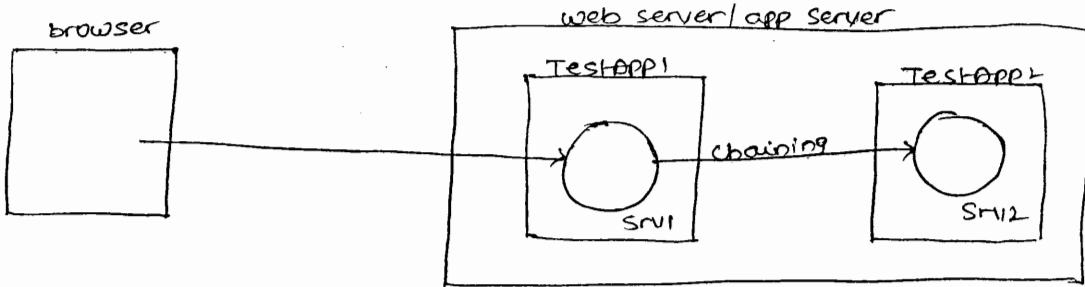
- (1) Invokable only on ServletContext object.
- (2) Expects logical name of the destination servlet/JSP program as argument value.
- (3) This method generated RequestDispatcher object can point only the destination servlet, JSP programs.

what is the difference between the RequestDispatcher object that is created based on RequestObject and ServletContext object?

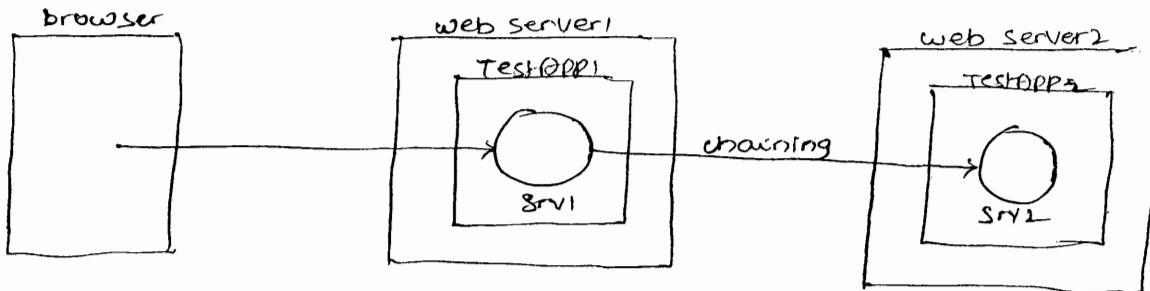
- (a) The Request object based RequestDispatcher object expects that source servlet program and destination web resource program in a same web application.
- \* The ServletContext object based RequestDispatcher object allows to keep the source servlet program and destination web resource program either in the same web application or in two different web applications of same server but they can not be there in two different web applications of two different servers.



Here srvt can use req obj or ServletContext obj based RequestDispatcher obj.



Here SRV1 should use `ServletContext` obj based `RequestDispatcher` obj

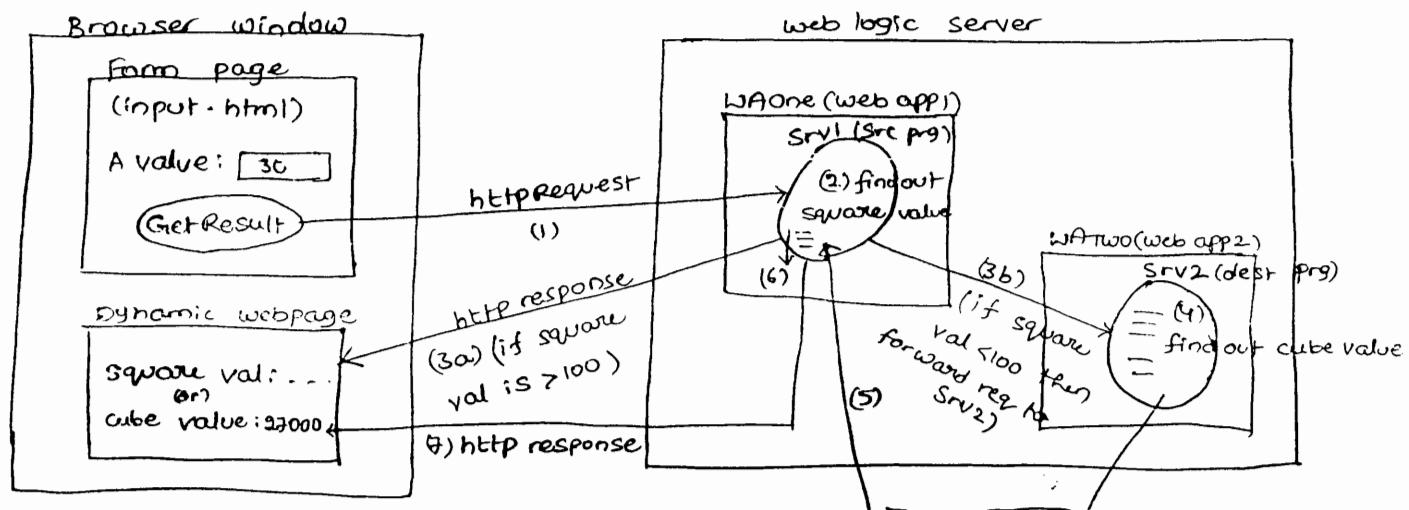


This kind of servlet chaining is not possible with `RequestDispatcher` obj

use send redirection concept

~~3/11/11~~ \* Servlet chaining is all about performing servlet-servlet communication.

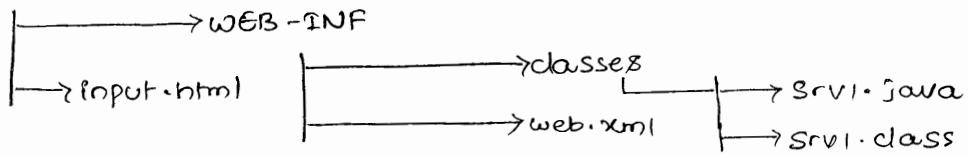
Performing servlet chaining between two servlet programs of two different web applications which reside in the same server



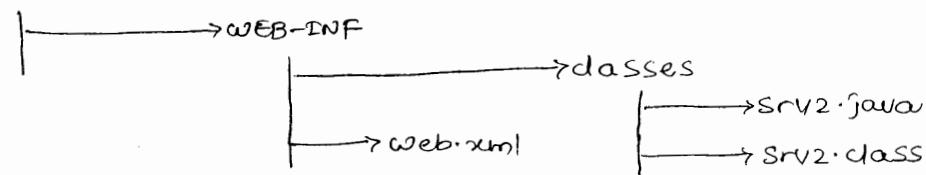
NOTE: In the above diagram SRV1 program forwards the request to SRV2 only when the generated square value is less than 100 otherwise the SRV1 directly sends response to browser window displaying that square value.

## Deployment directory structure

WAOne



WATwo



Deploy both these webapplications in web logic server (Adv.javaBatchDomain).

COPY WAOne, WATwo web applications to <oracle weblogic\_home>\user\_projects\domains\Adv.javaBatchDomain\autodeploy folder.

\* In SRV1 servlet program of the above WAOne webapplication we must create RequestDispatcher object based on ServletContext object.

### Source code (WAOne)

#### input.html

```

<form action="$1$url" method="get">
    ↴ url pattern of SRV1 servlet program
    A value: <input type="text" name="t1"><br>
    <input type="submit" value="getResult">
</form>

```

#### SRV1.java

```

//SRV1.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class SRV1 extends HttpServlet
{
    public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        PrintWriter pw = response.getWriter();
        response.setContentType("text/html");
    }
}

```

```

    //read form data
    int no = Integer.parseInt(req.getParameter("t1"));
    //find out square value
    int res = no * no;
    if (res >= 100) //display square value on browser window
    {
        pw.println("SRV1: square val is :" + res);
    }
    else
    {
        //forward the request to SRV2 program of WATWO web application
        //get Access to servlet Context obj of WAONE web application
        ServletContext sc1 = getServletContext();
        //get Access to servlet Context obj of WATWO web application.
        ServletContext sc2 = sc1.getServletContext("WATWO");
        //create RequestDispatcher obj pointing to SRV2 program of
        //WATWO web application
        RequestDispatcher rd = sc2.getRequestDispatcher("/S2url1");
        //forward the req to SRV2 program from SRV1 program
        rd.forward(req, res);
    }
}
//else
// service(--)
}

//class

```

### web.xml

Configure SRV1 program with /S1url as URL pattern.

### Source code (WATWO)

#### SRV2.java

```

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class SRV2 extends HttpServlet
{
    public void service (HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException
    {
        //general settings
        PrintWriter pw = res.getWriter();
    }
}

```

```

        res.setContentType("text/html");
        //read form data
        int no = Integer.parseInt(req.getParameter("H"));
        //find out cube value
        int res2 = no * no * no;
        //display cube value
        pw.println("SRV2: cube value is:" + res2);
    } // service()
} // class

```

### web.xml

Configure SRV2 program with the URL pattern /S2url

### URL to test the application

<http://localhost:7070/wAone/input.html>

\* In most of the servers the getContext() method of javax.servlet.ServletContext interface is not implemented so when that method is called in those servers we will get java.lang.NullPointerException. Due to this servlet chaining between two different servlet programs of two different web applications is not possible in most of the servers. (like weblogic 10.3, Tomcat all versions, JBoss versions, GlassFish all versions) but the above said chaining can be achieved in weblogic 8.x and 9.x servers.

\* Since getContext() method of servlet context interface is not implemented properly in all servers so we can say the RequestDispatcher object based servlet chaining is good only when both source servlet program and destination web resource program resides in same web application. In remaining situations it is recommended to use send redirection concept.

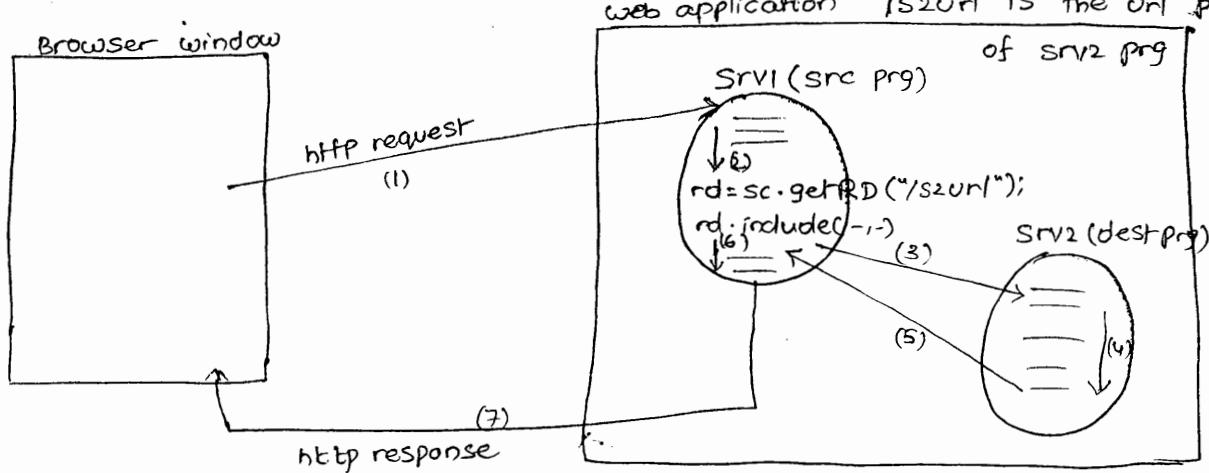
### What is the difference between rd.include(), rd.forward() method?

Ans Refer respective diagrams and key points of previous concepts.

\* To know the current java version >java -version

\* When multiple JDK softwares are installed in your computer having different versions then that JDK version will be activated whose <JdkHome>\bin directory is placed as first value of path environment variables.

## Understanding rd.include()



Here `Srv1` program (src prg), html o/p will not be discarded. moreover `Srv1`, `Srv2` servlet programs html outputs together goes to browser window as response.

\* `rd.include(...)` method performs including response mode of servlet chaining.

\* Both `Srv1` and `Srv2` programs will use same request and response object so the request data coming to `Srv1` is visible and accessible in `Srv2`. To pass additional data between `Srv1` and `Srv2` use request attributes.

\* Browser window gets response from `Srv1` program by having the html o/p of both `Srv1` and `Srv2` programs.

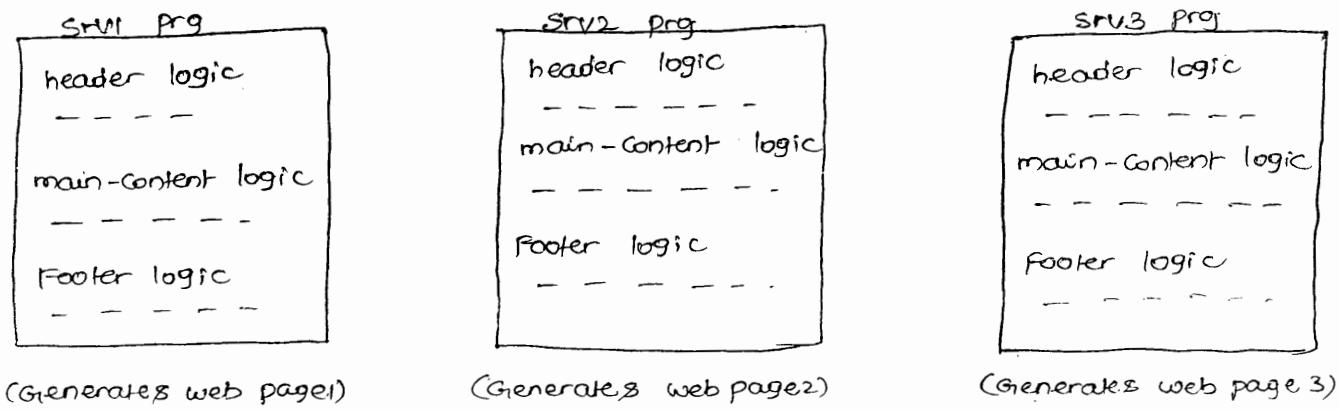
\* `Srv2` html o/p can be included anywhere in the html o/p of `Srv1` by calling `rd.include(...)` method in the required place.

\* `Srv1` and `Srv2` can be there in the same web application (or) can be there in two different web applications of same server.

\* `Srv2` can be a servlet program, JSP program or html program.

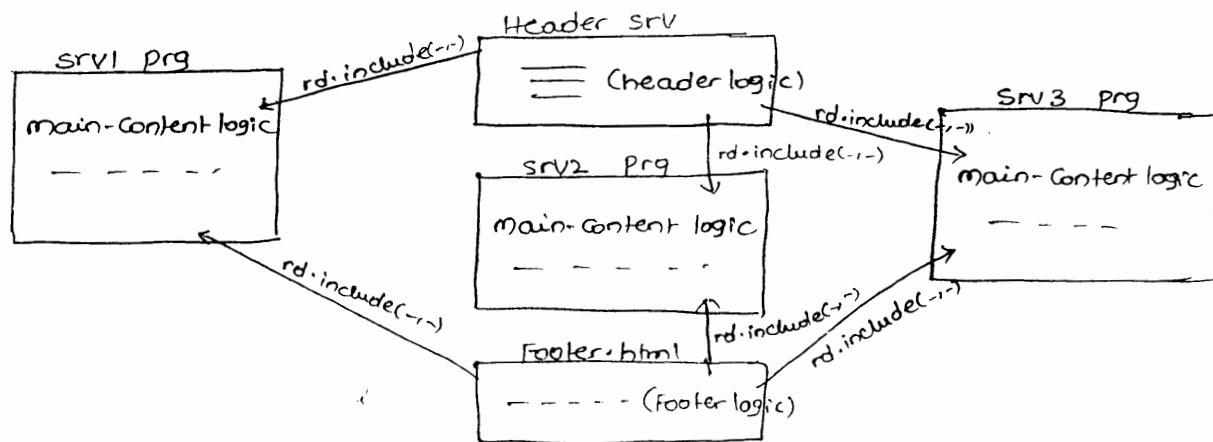
\* In real time `rd.include(...)` is useful to include the o/p of reusable logics like header and footer logics in the output of (html output) main web resource programs.

## problem :



- \* All web pages of website generally contains same header and footer contents but the main content will be changed in every web page.
- \* In the above diagram header and footer logics are not reusable because they are placed in all the three servlet programs even though they are same for all the three servlet programs.

## solution :



- + In solution diagram header and footer logic are separated from main servlet programs of web application and they are placed in two separate web resource programs but their html output is included in main servlet using rd.include(<-->). This indicates header and footer logics are made as reusable logics.

sample code based on solution diagram

Headersrv.java (having header logic)

public class Headersrv extends HS

{

    public void service(--) throws ServletException, IOException

{

        PrintWriter pw = res.getWriter();

        res.setContentType("text/html");

        pw.println("<font color=red size=6> S A T H Y A </font>");

        pw.println("<br><br><br><hr>"); //Don't place pw.close() in

}

    } /headerurl is the Headersrv servlet program

Headersrv.

Footer.html (having footer logic)

<hr>

<br><br><br>

<b><i> © All rights reserved 2101-11 </b><i>

MainServlet prgs

Public class TestSrv1/2/3...In extends HS

{

    public void service(--) throws SE, IOE

{

        RequestDispatcher rd1=null, rd2=null;

        try

{

            //include header Content from Headersrv

            rd1=req.getRequestDispatcher("/headerurl");

            rd1.include(req,res);

            //logic to generate main Content of web page

            //include footer content from Footer.html

            rd2=req.getRequestDispatcher("/footerurl");

            rd2.include(req,res);

        }

    catch (Exception e)

{

        e.printStackTrace();

}

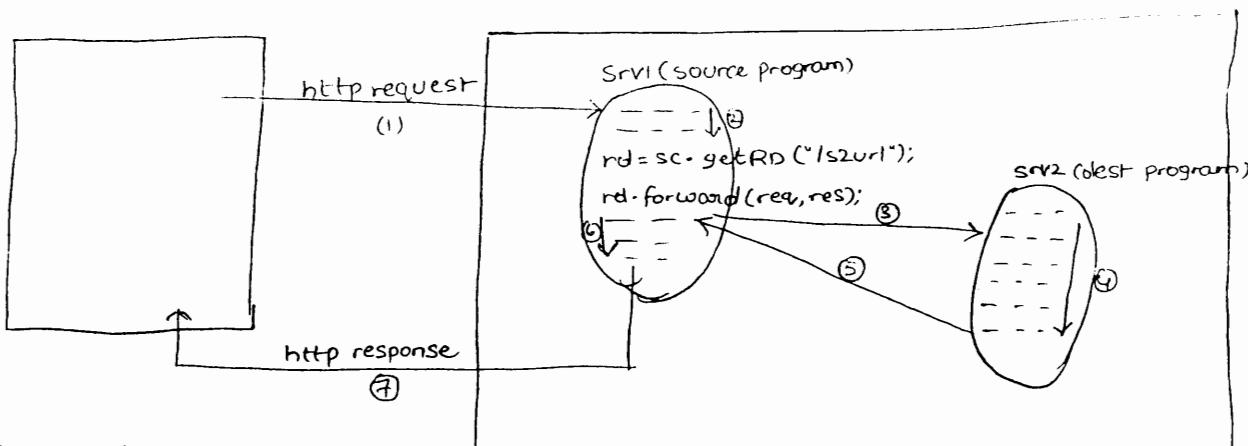
}

}

- \* Once the `rd.forward(...)` method executed in source servlet program then the effect of `rd.include(...)` method call will not be there in that servlet program that means `rd.forward(...)` method call discards both direct and included html outputs of source servlet program.
- \* For example application on both `rd.forward(...)`, `rd.include(...)` method calls refer app (6) of the page no 67-70.
- \* ~~Also~~ For example application on `rd.forward()` and `rd.include()` also refer app (8) of material.
- \* While working with `rd.include(...)` method don't commit the response in destination web source program by calling `pw.close()` method because it won't allow us to add further output generated by source servlet program and other destination programs. But we can do this work while working with `rd.forward(...)` method.
- \* Don't commit the response in source servlet program by calling `pw.close()` method before `rd.forward(...)` method call because this process throws `java.lang.IllegalStateException` (can't forward after response has been committed).

NOTE: `pw.close()` commit the response of web resource program that means it does not allow to add further content to response.

### Understanding `rd.forward(...)` method



The html output of `SRV1`

will be discarded and only the html output of `SRV2` will go to `http response`.

### key points:

- \* `rd.forward(--)` method is there to perform forwarding request mode of servlet chaining.
- \* Both `srv1` and `srv2` servlet programs will use same request and response objects so the request data coming to `srv1` (lik request parameters, headers are visible and accessable in `srv2` program).
- \* To pass additional data between `srv1` and `srv2` programs use "request" attributes.
- \* All the statements placed in `srv1` program before and after `rd.forward` method will be executed but the entire html output of `srv1` program will be discarded.
- \* Both `srv1` and `srv2` can be there either in the same webapplication or two different webapplications of same server.
- \* `srv2` can be a servlet program or JSP program or html program.
- \* `rd.forward(--)` is very useful to configure Error Servlet for other main servlet programs of the web application.

### what is Error servlet?

- (A) The servlet program that executes only when exceptions are raised in other servlet program is called as Error servlet. This servlet is useful to display exception related messages as non-technical guiding messages on browser window when exceptions are raised in other servlet programs of web application.

### sample code to configure Error Servlet

#### ErrSRV.java

```
public class ErrSRV extends HttpServlet
{
    public void service(--) throws SE, IBE
    {
        res.setContentType ("text/html");
        PrintWriter pw = res.getWriter();
        pw.println ("<font color=red>Internal Problem </font>");
    }
}
```

```
pw.println("<br> Go to <a href = \"input.html\"> Home </a>");  
} // service(-,-)  
  
} // class  
// "/eurl" is the url pattern of ErrSrv program (in web.xml)
```

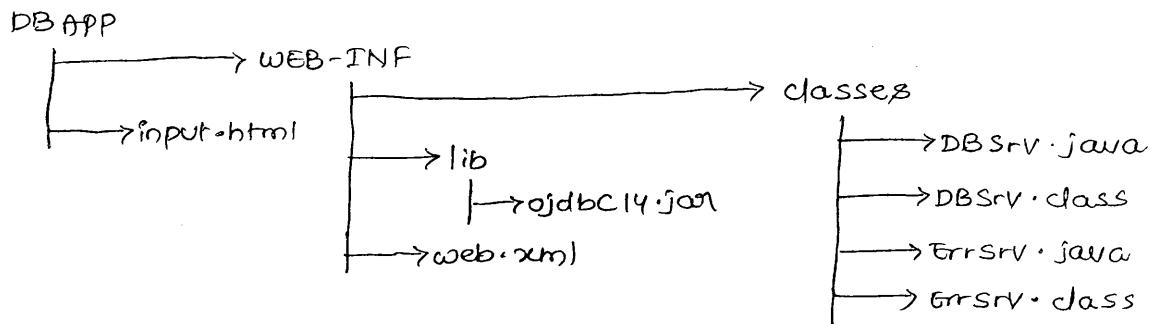
### DBSrv.java (main servlet program of web application)

```
public class DBSrv extends HttpServlet  
{  
    public void init()  
    {  
        ---  
        ---  
        --- Same as previous application  
        --- (JDBC code)  
        ---  
        ---  
    }  
    public void doGet (-,-) throws ServletException, IOException  
    {  
        try  
        {  
            ---  
            --- Same as previous application  
            --- (JDBC code)  
        }  
        catch (Exception e)  
        {  
            try  
            {  
                RequestDispatcher rd = req.getRequestDispatcher ("/eurl");  
                rd.forward (req, res);  
            }  
            catch (Exception e1)  
            {  
                e1.printStackTrace();  
            }  
        }  
    }  
}
```

```
public void doPost (...) throws SE, IOE  
    {  
        doGet(req,res);  
    }  
public void destroy()  
{  
    -- -- --  
}  
}
```

- \* In the above sample code when exception is raised in the doGet(...) method of DBSRV Servlet Program the Error servlet (ErrSrv) program will be executed automatically because of the rd.forward (...) method call.
- \* If servlet program contains rd.forward (...) method call then that servlet programs html output will be discarded but their s.out(" ") statements output will not be discarded.
- \* Always keep rd.forward (...) method call in source servlet program with condition statement show that the html output of that source servlet program will be discarded only when that condition is satisfied otherwise the html output will not be discarded.
- \* In real time projects rd.forward (...) useful to we Configured Error servlet and to make web applications as more end user friendly web applications.

Deployment directory structure of the above sample code based web application

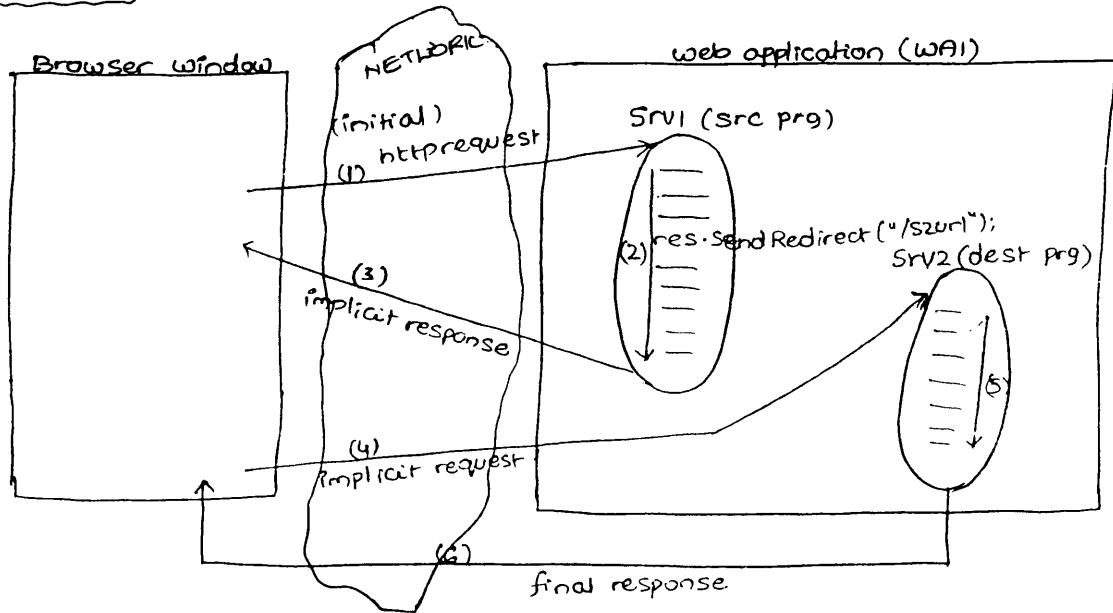


\* Develop the above web application resources based on above sample code and application ④ of the booklet.

**WTF!**  
\* The limitation with RequestDispatcher object based servlet chaining is it can not be used when source servlet program and destination web resource program are placed in two different web applications of same server (very few servers are supporting this) (or) in two different web applications of two different servers. To overcome this problem use send redirection concept of response.sendRedirect() method.

### Understanding sendRedirection

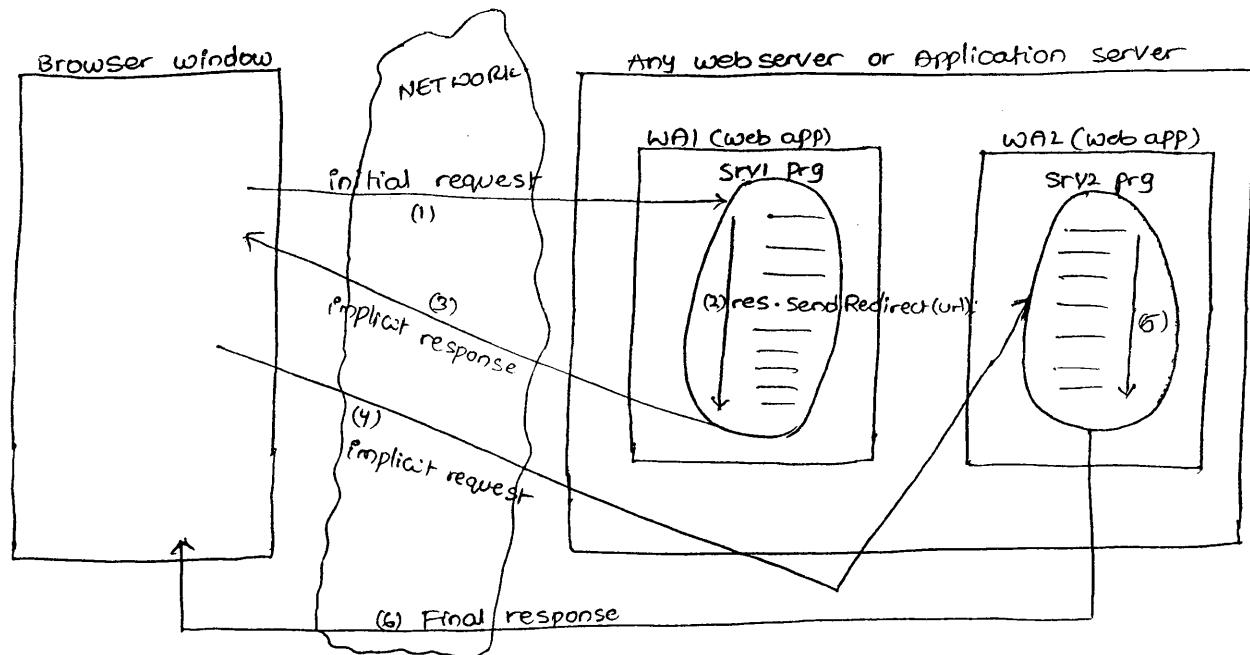
#### Diagram (1)



The html output of SRV1 (src) servlet program will be discarded and only the html output of SRV2 (dest) servlet program will be displayed on browser window as response.

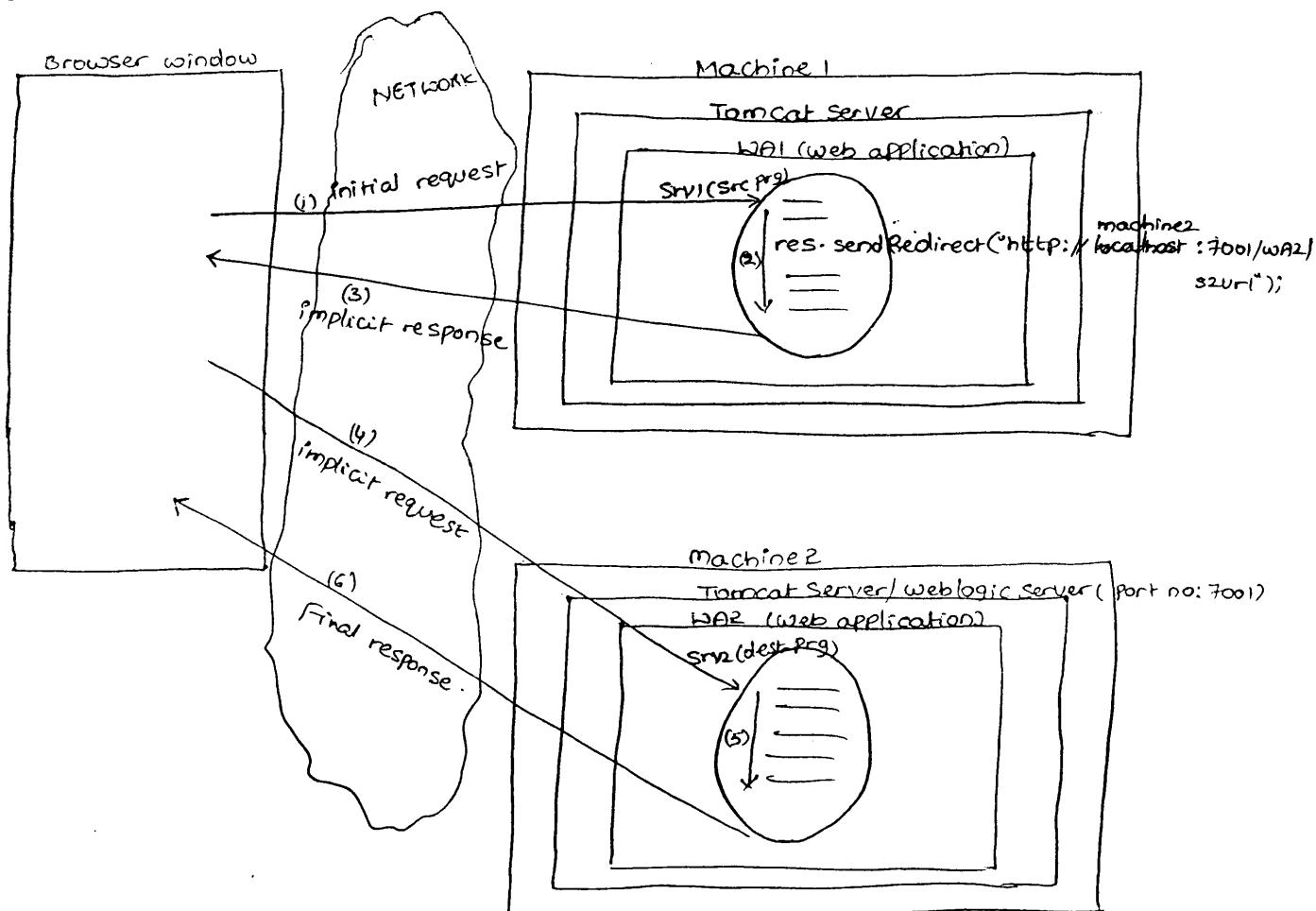
/Srv2 is the URL pattern of SRV2 servlet program.

Diagram(2)



url : `http://localhost:2020/WA2/s2url` (Request url of **Srv2** program).

Diagram(3)



\* Using send redirection concept we can not perform response inclusion but we can forward request from one web servlet program to destination web resource program.

\* With respect to the diagrams

(1) Browser window gives initial request to SRV1 servlet program.

(2) All the statements of SRV1 servlet program executes including res.sendRedirect(-) method.

(3) SRV1 program generates implicit response to browser window having the URL placed in sendRedirect(-) method as argument value. The response status code of this implicit response is 300-399 (indicates redirection).

(4) Browser window uses the URL coming from implicit response (because of 300-399 status code) and generates implicit request to SRV2 program.

(5) All the statements of SRV2 program executes.

(6) The output of SRV1 program will be discarded and only the HTML output of SRV2 program goes to browser window as final response.

#### Key points

\* SRV1 and SRV2 will not use same request and response objects. They use different sets of req, res objects. So the request data coming to SRV1 program is not visible and accessible in SRV2 program.

\* To send additional data from SRV1 program SRV2 program append query string to the URL of res.sendRedirect(-) method in SRV1 program and read those values in SRV2 program as request parameter values.

#### In SRV1 Prg

```
res.sendRedirect("/s2url?p1=val1&p2=val2");  
(or)
```

```
res.sendRedirect("http://localhost:2020/wa2/s2url?p1=val1&p2=val2");  
(or)
```

```
res.sendRedirect("http://machine2:7001/wa2/s2url?p1=val1&p2=val2");
```

#### In SRV2 Prg

```
String s1 = req.getParameter("p1");  
String s2 = req.getParameter("p2");
```

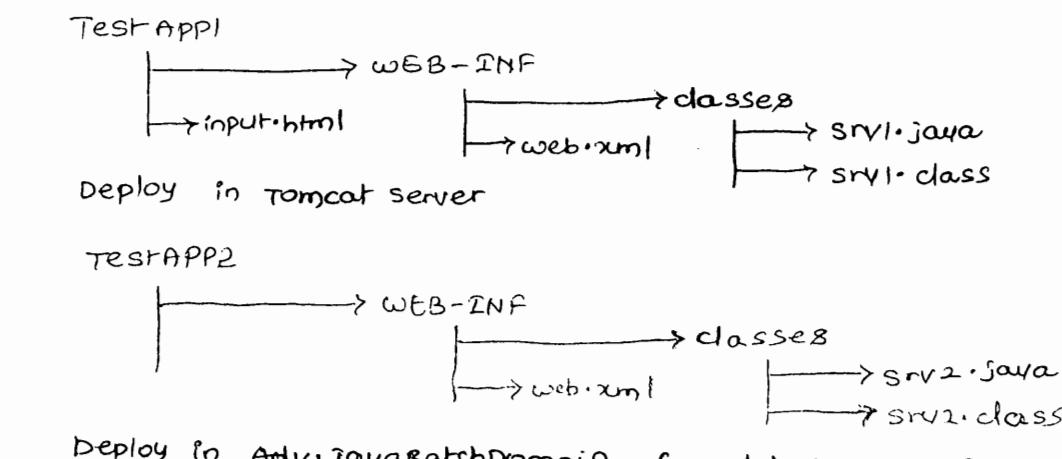
- \* SRV1 and SRV2 can be there in the same web application or can be there in two different web applications of same server or two different servers (these two servers can be there in the same machine or in two different machines).
- \* If SRV1 and SRV2 resides in the same web application we can pass relative path in sendRedirect(-) method otherwise we must pass absolute URL.
- \* SRV2 can be a servlet program or JSP program or html program or ASP program or ASP-.net program or PHP program etc...
- \* The movement res.sendRedirect(-) method executes in SRV1 program the entire html output of SRV1 program will be discarded.
- \* In real time to pass the request of one website to another website without worrying about their technology environment we can use send redirection.

Ex: IBM has acquired rational.com so the request given to rational.com will be redirected to certain web resource program of ibm.com.  
 Oracle Corporation has acquired sunmicrosystems so the request given to sun.com will be redirected to a webresource program of oracle.com.

#### Example application on send Redirection

NOTE: In the below application SRV1 program is sending three values to SRV2 program by appending query string to the url of response.sendRedirect() method. Those values are form data, sum result.

#### Directory structure



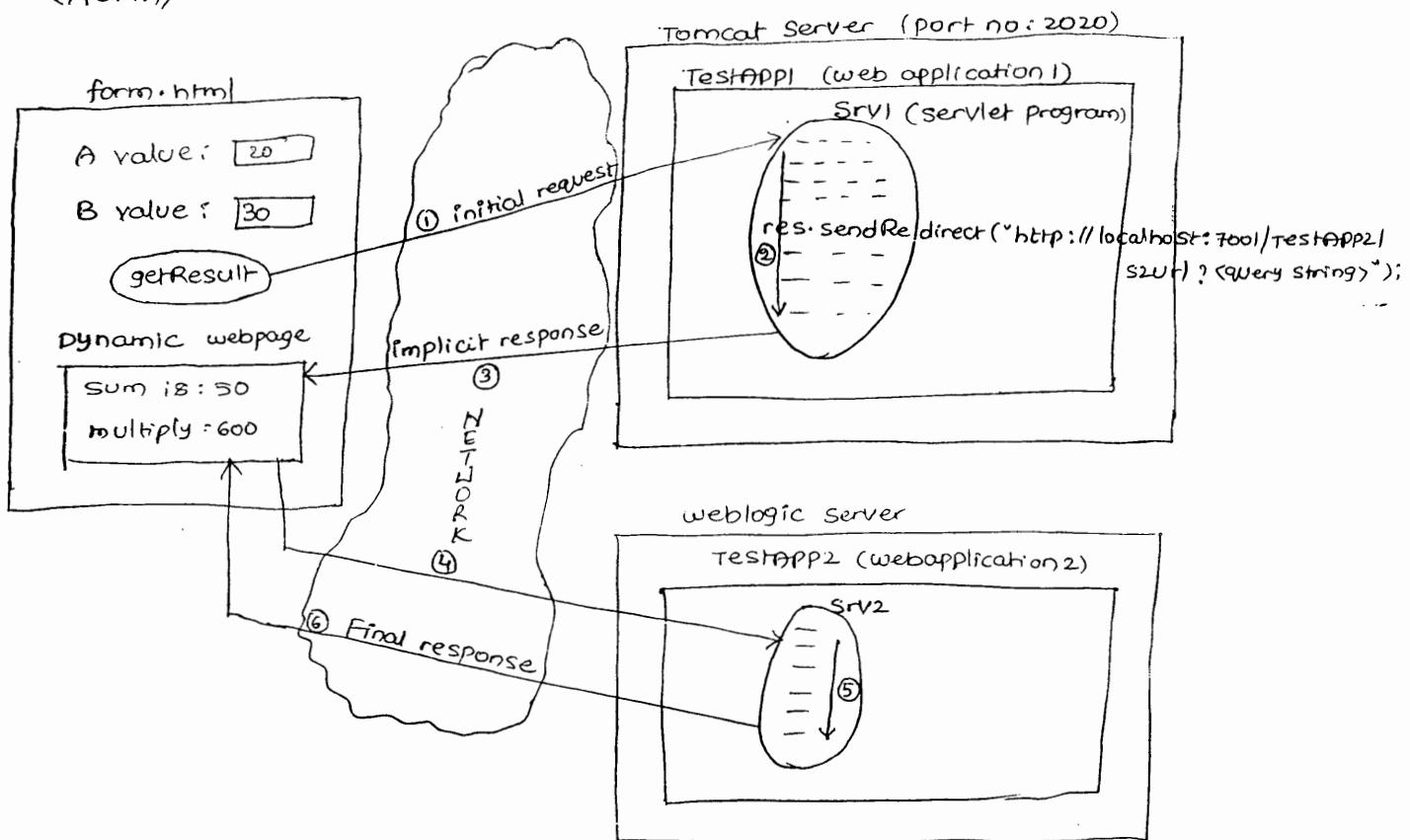
## TestAPP1 (web application)

### input.html

```

<form action = "s1url">
    A value : <input type = "text" name = "t1"> <br>
    B value : <input type = "text" name = "t2"> <br>
    <input type = "submit" value = "getResult" />
</form>

```



### Srv1.java

```

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

public class Srv1 extends HttpServlet {
    public void service (...) throws SE, IOException {
        // general settings
        PrintWriter pw = res.getWriter();
        res.setContentType ("text/html");
        // read form data
        int val1 = Integer.parseInt (req.getParameter ("t1"));
    }
}

```

```

    int val2 = Integer.parseInt(req.getParameter("t2"));
    //find sum value
    int sum = val1 + val2;
    pw.println("<b> srv1 : sum is </b>" + sum);
    //redirect the request to srv2 program of testapp2
    s.o.p("srv1 : before res.sendRedirect(-)");
    res.sendRedirect("http://localhost:7001/testapp2/s2url ? P1=" + val1 + " & P2=" +
    " + val2 + " & P3=" + sum);
    s.o.p("srv2 : after res.sendRedirect(-)");
    //close stream obj
    pw.close();
}
//service(-,-)

```

query string having data to send data to srv2 program.

3) class

web.xml

Configure Srv1 server program with "/s1url" url-pattern.

TESTAPP2 source code (web applicationz of different server)

Srv2.java

```

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class srv2 extends HttpServlet
{
    public void service(-,-) throws SE, IOException
    {
        //general service
        PrintWriter pw= res.getWriter();
        res.setContentType("text/html");
        //read request param values send Srv1 program
        int val1 = Integer.parseInt(req.getParameter("P1"));
        int val2 = Integer.parseInt(req.getParameter("P2"));
        int sum = Integer.parseInt(req.getParameter("P3"));
        //find out multiply value
        int mul = val1 * val2;
        //display results
        pw.println("srv2 : sum is :" + sum);
        pw.println("srv2 : multiply is :" + mul);
        s.o.p("from Srv2 Program");
        pw.close();
    }
}

```

## web.xml

configure SRV2 program with "/s2url" url-pattern.

Request URL to test the application

http://localhost:2020/testapp1/form.html

what is the difference between rd.forward() and res.sendRedirect() ?

rd.forward(-,-)

res.sendRedirect(-,-)

- \* perform forward mode of servlet chaining.
- \* The source servlet program communicates with destination webresource program directly.
- \* The source servlet program and destination web resource program uses same request and response objects So request data coming to source servlet program is visible and accessible in destination webresource program.
- \* Source servlet program can use request attributes to send additional data to destination program.
- \* Source servlet and destination programs can be there in same web application (any server) (or) can be there in two different web applications of same server (only in few servers).
- \* Destination program can be a servlet (or) JSP (or) html program.
- \* During forwarding request operation url in the browser address bar will not be changed.
- \* suitable when source servlet program and destination servlet program resides in the same web application.

\* performs sendRedirection mode of communication.

\* The source servlet program communicates with destination servlet program by having network round trip with browser window.

\* The source servlet program and destination web resource program will not use same request and response object so the request data coming to source servlet program is not visible and accessable in destination program.

\* Append query string to the url of response.sendRedirect() method to send additional data from source servlet program to destination program.

\* The source servlet program and destination program can be there in same web application (or) in two different web applications of same server (or) different servers.

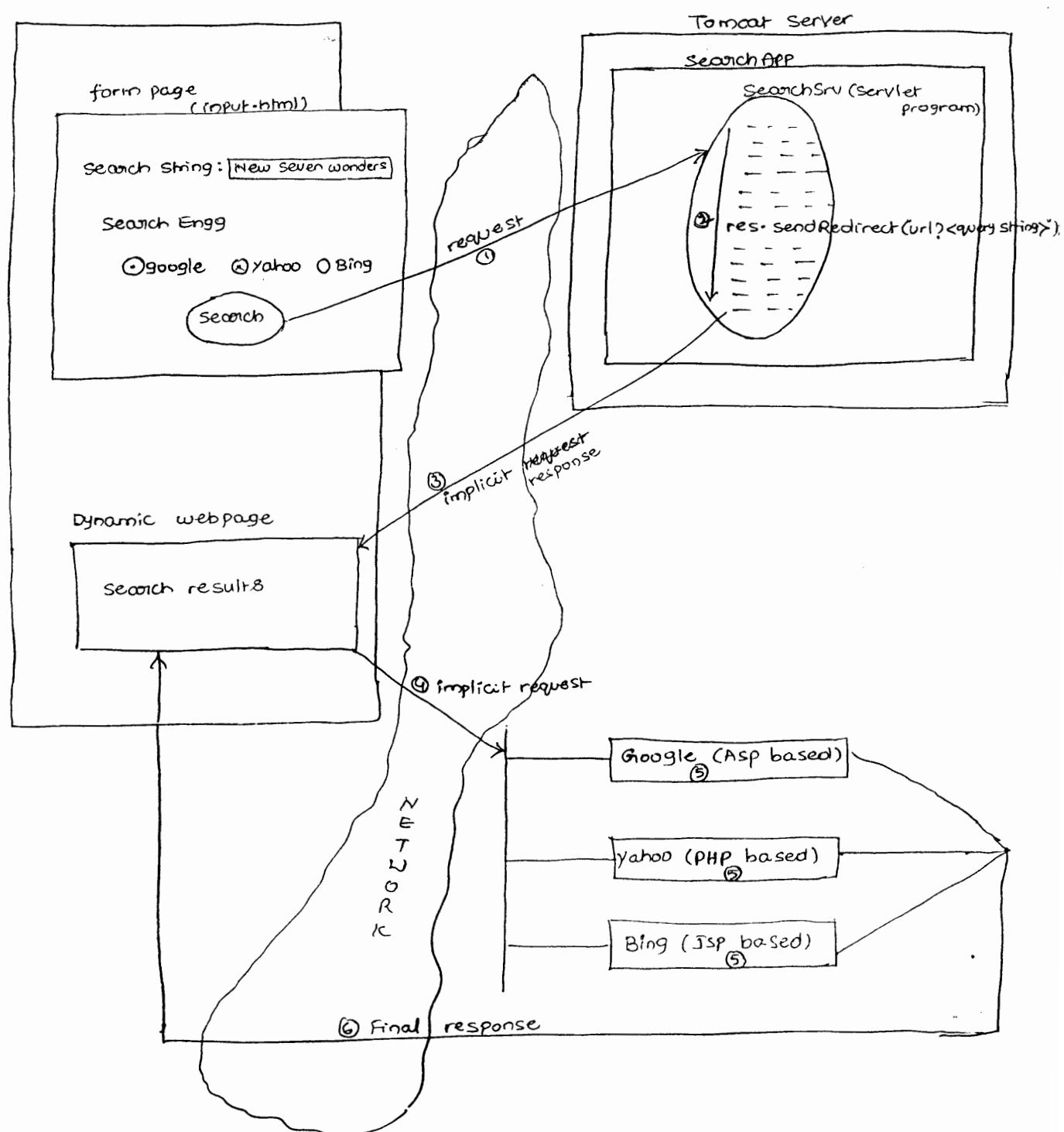
\* Destination program can be a servlet (or) JSP (or) html (or) ASP (or) ASP.net (or) PHP programs etc...

\* During sendRedirection operation the url in the browser address bar will be changed.

\* suitable when source servlet program and destination program resides in the two different web applications.

## Example on send Redirection :

Servlet program redirects the request to non Java web resource programs of Internet websites.



- \* The above application act as hub for working with multiple search engines.
- \* The above application the servlet SearchSrv redirect the request to search engine based on the search engine that is chosen in the form page.

\* To work with above application gather the web resource program names and request parameter names of different search engines as shown below.

For google

http://www.google.co.in/search ? q = new + seven + wonders

request parameter value

For yahoo

http://search.yahoo.com/search ? p = new + seven + wonders

For bing

http://www.bing.com/search ? q = new + seven + wonders

web resource program name

\* For exact source code of above diagram refer application (9) of the page numbers 73-75.

How many ways are there to pass data between the source servlet program and destination web resource program?

(A) If source servlet program and destination web resource program resides in same web application then use

- (1) request attributes
- (2) session attributes
- (3) servletContext attributes

If source servlet program and destination web resource program resides in two different web applications of same/different server(s).

(1) append query string to URL of res.sendRedirect() method

```
res.sendRedirect("url ? <query string>");
```

Ex.

```
res.sendRedirect("http://localhost:7001/testapp2/s2url ? p1=10 & p2=20");
```

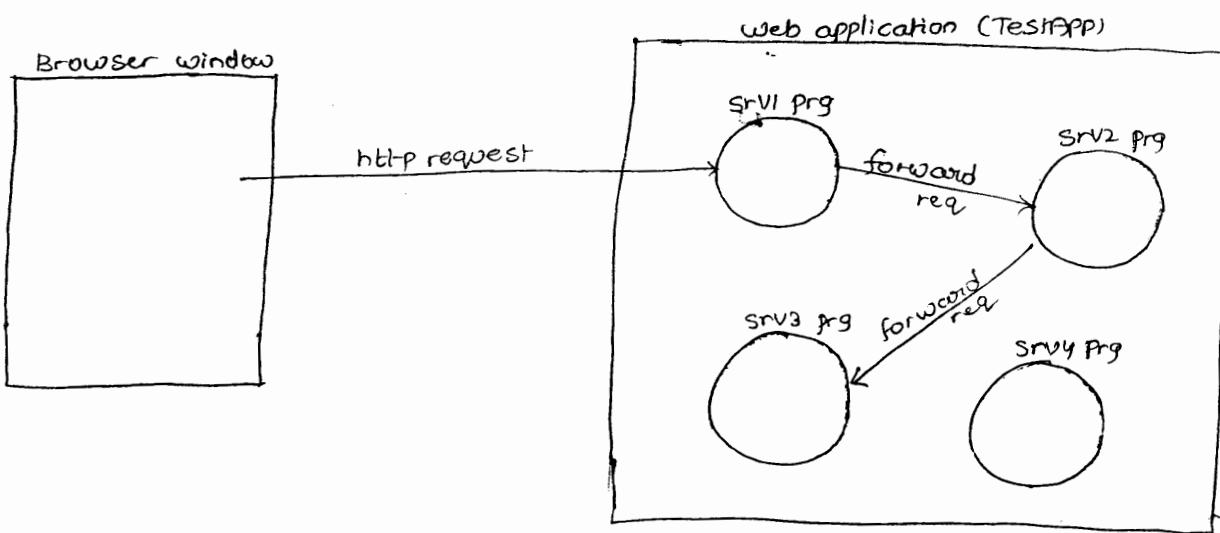
\* Attribute is a special logical name that can hold java object as an value.

\* Attributes provide more visibility and accessibility to the data of the application compare to the variables declared in servlet program.

\* Attributes allows us to store only objects. If we give simple values to attributes then they will be converted into wrapper class objects and will be stored as attribute values.

### Request attributes

\* These attributes allocate memory in request object and they are visible through out request cycle.



\* Request attributes that are created <sup>in</sup> one servlet program of servlet chaining are visible in other servlet programs of same servlet chaining because all the servlet programs of a servlet chaining will use same request and response objects.

\* In the above diagram the request attribute created in srV1 program is visible and accessible in srV2, srV3 program but not visible in srV4 program. because srV1, srV2, srV3 programs are using same request and response objects by participating in servlet chaining and srV4 program is not participating in that servlet chaining.

### To create request attribute

```
req.setAttribute("name", "raja");
attribute name → attribute value
req.setAttribute("age", new Integer(50));
req.setAttribute("total", 600);
attribute value will be converted into wrapper class object Integer
through autoboxing.
```

### To modify request attributes values

```
req.setAttribute("name", "raja");
req.setAttribute("age", new Integer(45));
req.setAttribute("total", 550);
```

NOTE: The method `setAttribute(...)` can create new request attribute or can modify existing request attribute value.

To read values from request attributes

```
String s1 = (String)req.getAttribute("name");
```

```
Integer s2 = (Integer)req.getAttribute("age");
```

```
int s3 = req.getAttribute("total");
```

J- converts wrapper class object Integer to simple int through auto unboxing concept.

To delete/remove request attribute

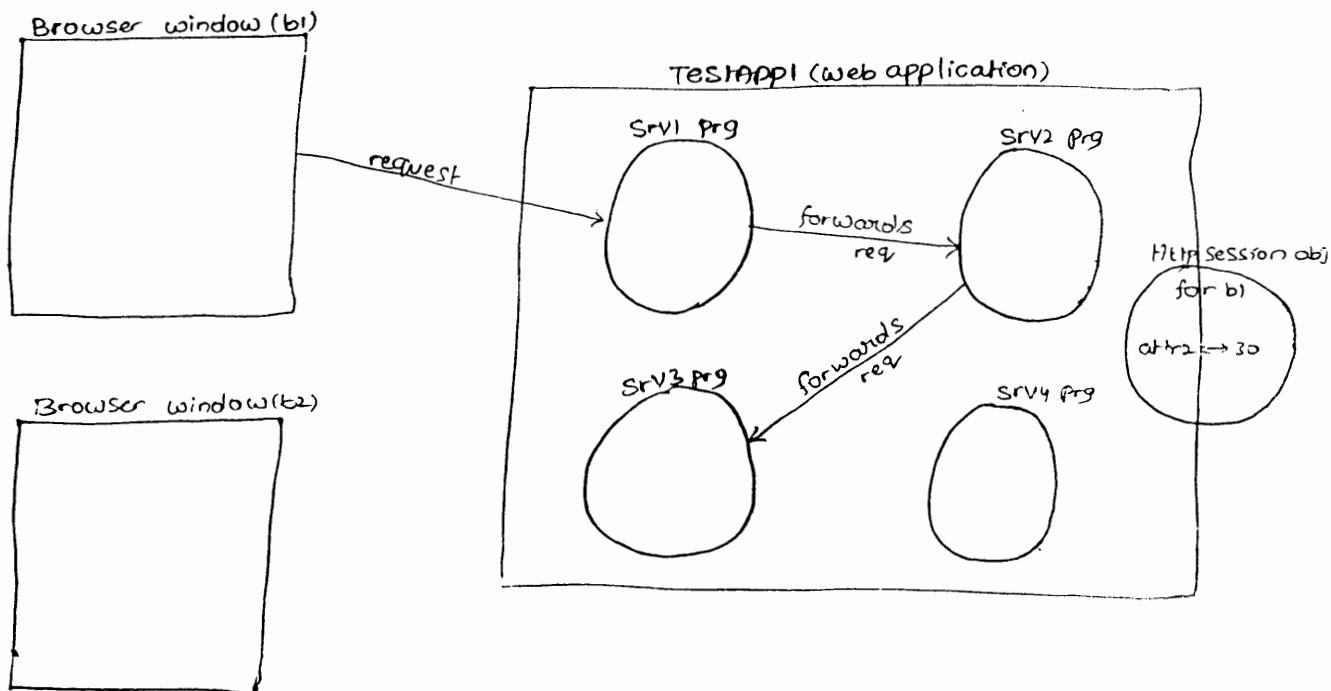
```
req.removeAttribute("name");
```

```
req.removeAttribute("age");
```

```
req.removeAttribute("total");
```

### Session Attribute

\* Session attributes allocate memory in http session object. ~~so~~ The http session object allocates memory on the server and it is one per browser window so the session attributes of http session object are visible and accessible in all web resource programs of web application irrespective of their request and response objects but they must get request from that browser window for which http session object and its attributes are created.



\* The session attribute created in servlet program by getting request from browser window is visible and accessible in all other servlet programs of web application but they must get request from same browser window.

\* Session attributes are global attributes in the web application but they are specific to a browser window (client).

To create / access HttpSession object for browser window on servlet

```
HttpSession ses = req.getSession();  
                  ↓  
                  creates or locates HttpSession object
```

NOTE: This req.getSession() method checks the availability of session object on server for browser window, if available it provides access to that session object otherwise (if not available) it creates new session object (HttpSession object) for browser window on the server.

Session attributes

To create ses attribute or to modify ses attribute value

```
ses.setAttribute("attr2", new Integer(30));
```

To read ses attribute value

```
String s1 = (String) ses.getAttribute("attr2");  
Integer i1 = Integer
```

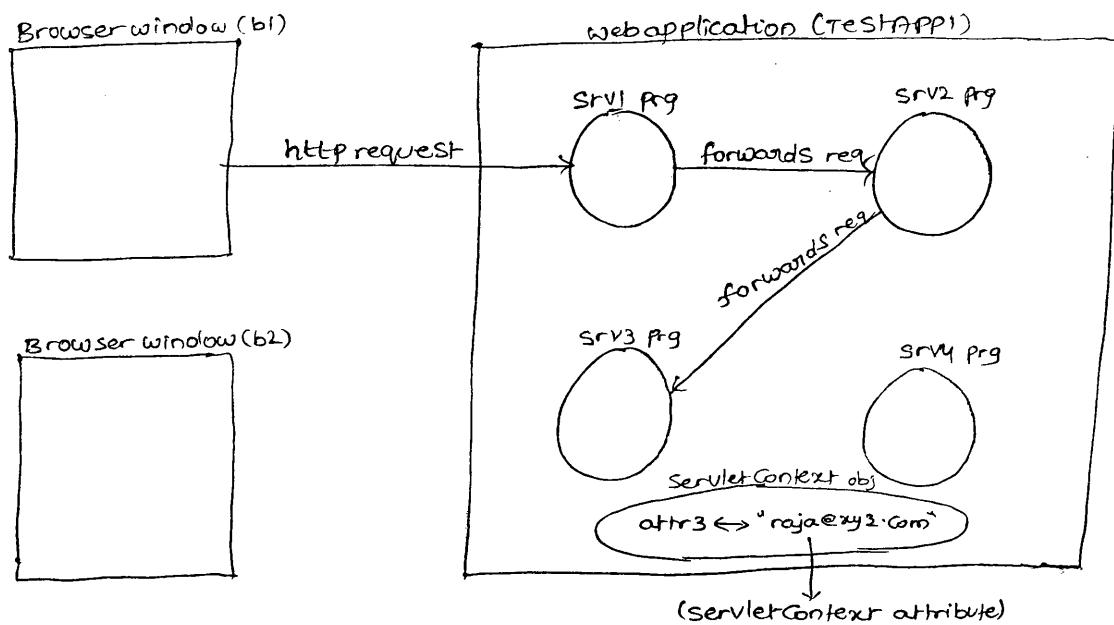
To remove ses attribute

```
ses.removeAttribute("attr2");
```

ServletContext attributes

+ ServletContext attributes allocate memory in servletContext object. Since servlet context object is one per web application and global memory of web application so the servletContext attributes are visible and accessible in all web-resource programs of web application irrespective of request and response objects they are using and irrespective of the browser window from which they are getting request.

+ In the below diagram the servletContext attribute created in servlet program is visible and accessible in all the remaining servlet programs of web application irrespective of the request and response objects they are using and irrespective of browser window from which they are getting request.



TO create or modify servletContext attribute

```
ServletContext sc = getServletContext(); // gives access to ServletContext object
sc.setAttribute("attr3", "raja@xyz.com");
```

TO read servletContext attribute value

```
String s1 = (String) sc.getAttribute("attr3");
```

TO remove servletContext attribute

```
sc.removeAttribute("attr3");
```

what is the difference between request parameters and request attributes?

(A) Request parameters are client supplied input values to web resource program (form data). Request attributes are programmer created additional data to pass from one web resource program to another web resource program when they are participating in chaining (like servlet chaining).

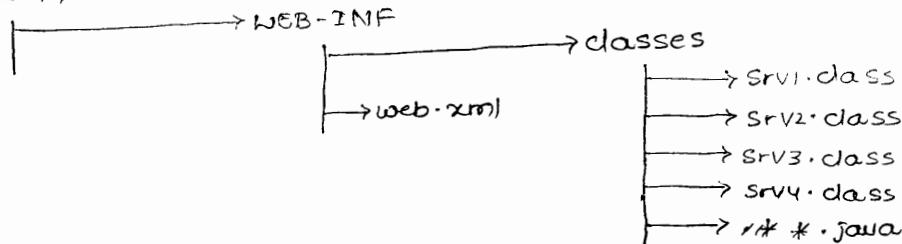
### Conclusion

- \* If source servlet program and destination program resides in the same request and response objects then use request attributes to send the data.
- \* If source servlet program and destination program are getting request from some browser window then we use session attributes for sending data.

\* If source servlet program and destination webresource program can not satisfy above two conditions then use ServletContext attributes for sending data.

Example application on to understand the scope of various attributes

### Attributes APP



### Srv1.java

```
public class Srv1 extends HttpServlet
{
    public void service(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
    {
        //general settings
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        //creating request attribute
        req.setAttribute("attr1", "raja");
        //creating session attribute
        HttpSession ses = req.getSession(); //creates session obj
        ses.setAttribute("attr2", new Integer(30)); // creates session attribute
        //creating servletContext attribute
        ServletContext sc = getServletContext(); //creates session obj
        sc.setAttribute("attr3", "raja@xyz.com"); // creates servletContext attribute
        //forward req to Srv2 prg
        RequestDispatcher rd = req.getRequestDispatcher("/Srv2");
        rd.forward(req, res);
    } //service()
} // class
```

### srv2.java

```

public class srv2 extends HttpServlet
{
    public void service (...,...) throws SE,IOE
    {
        //general settings
        res.setContentType ("text/html");
        PrintWriter pw = res.getWriter();
        //reading req attribute value
        pw.println ("srv2: attr1(request) attribute value:" + req.getAttribute ("attr1"));
        //reading session attribute value
        HttpSession ses = req.getSession(); //gives access to session obj
        pw.println ("srv2: attr2(session) attribute value:" + ses.getAttribute ("attr2"));
        //reading servletContext attribute value
        ServletContext sc = getServletContext();
        pw.println ("srv2: attr3 (servletContext) attribute value:" + sc.getAttribute ("attr3"));
        //forward req to srv3 Prg
        RequestDispatcher rd = req.getRequestDispatcher ("~/srv3");
        rd.forward (req, res);
    } //service(...)

} //class

```

### srv3.java

same as srv2.java and don't forward request to any other servlet.

### srv4.java

same as srv3.java

To test the above application give first request to srv1 program and give other than first request from to other than srv1 program from same or different browser windows and check the visibility of attributes.

http://localhost:2020/AttributesApp/srv1

http://localhost:2020/AttributesApp/srv2

http://localhost:2020/AttributesApp/srv3

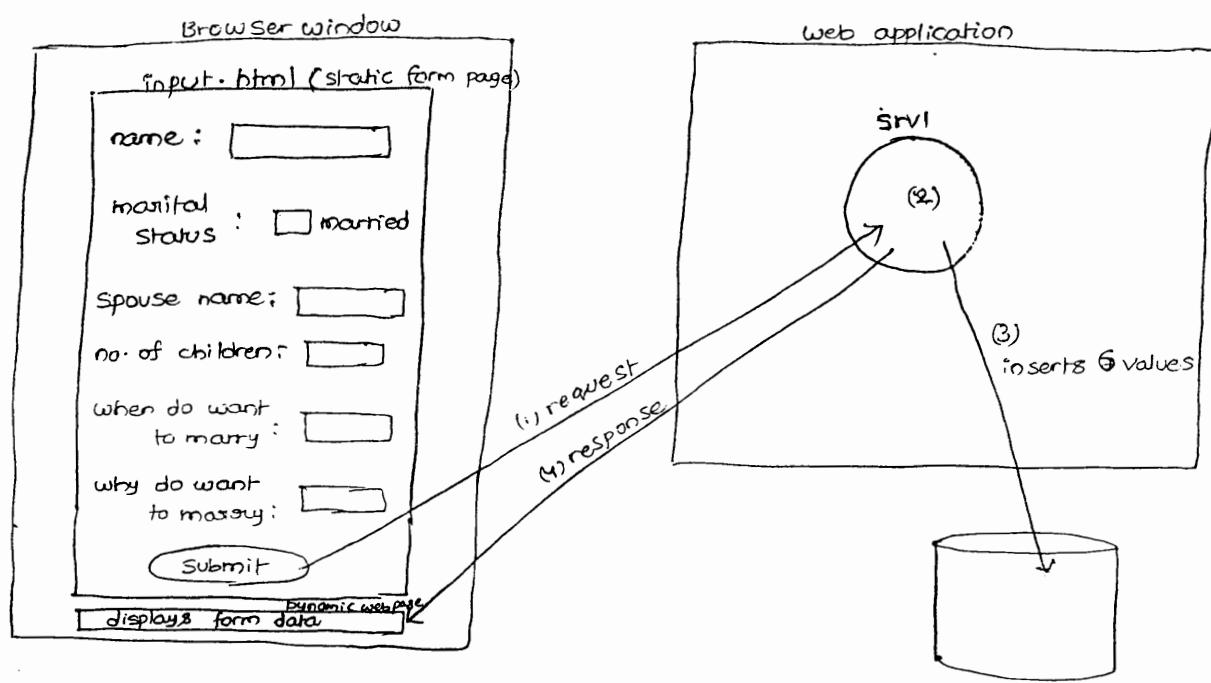
http://localhost:2020/AttributesApp/srv4

### web.xml

Configure srv1 servlet program with /srv1 program url pattern.

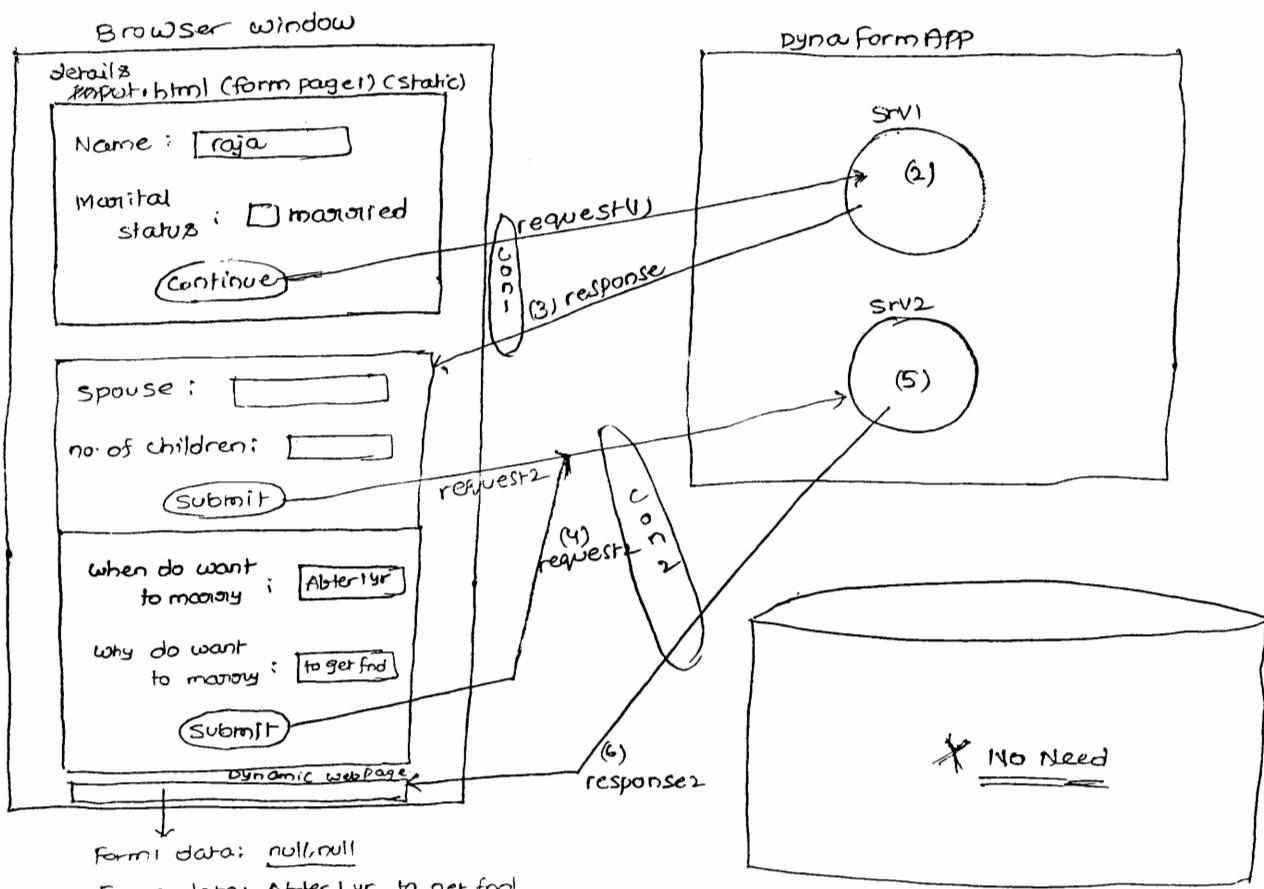
Srv2	/srv2
Srv3	/srv3
Srv4	/srv4

- \* The form page that is prepared through .html file is called as static form page (fixed content).
- \* The form page that comes as response generated by servlet or JSP program is called as dynamic form page (dynamic content).
- \* Instead of asking all the details of end user in a single form page it is recommended to ask those details in multiple form pages having the support of dynamic form pages.



bad designing of form page

- \* In the above static form page the end user needs to read and answer some unnecessarily questions when he selects or deselects the marital status check box. To overcome that problem work with dynamic form page as shown below.
- \* In the below diagram the content of form page 2 is dynamic and is regenerated by SRVI program based on the marital status value of form 1 or request1 data.
- \* All web applications are stateless web applications by default that means they can not remember client data across the multiple request during a session.
- \* A session is a set of continuous and related operations (request) performed on the web application by a client (browser window/end user).  
Ex: sign-in to sign-out in gmail.com, start of the class to end of a class on a particular day.



Good form Designing but web application is stateless here that means while processing request2 in srv2 program we can't use request1 data (form1 data).

\* The stateless behaviour of web application is nothing but while processing current request in any web resource program we can't use previous request data that means while processing request2 we can't use request1 data. Similarly while processing request3 we can't use request1 and request2 data as shown in the above diagram.

\* web applications are stateless because the protocol http is given as stateless protocol. According to this one new connection will be created between browser window and web server for every request and this connection will be closed automatically once request related response goes to browser window.

Ex: If browser window gives 10 requests to a web application then 10 connections will be created between browser window and webserver, due to this one connection related client's request data can not be used in other connection related client's request processing (that means we can't use previous request data in web resource program while processing current request).

- \* For the above diagram based application refer application(10) of page no. 74-76
- \* By placing <form> tag, <input> tag in pw.println() statements of Servlet program we can generate dynamic form page from that servlet program. (refer Srv1.java of page No.75).
- \* In naukari.com registration process multiple forms will be there. In that first form page is static form page and other form pages are dynamic form pages. In that based on the data given in form page1 the questions in form page2 will be rendered. similarly base on data in form page2 the question the questions in form page3 will be rendered.
- \* web applications are stateless because they are using a stateless protocol called Http.

Why HTTP is designed as stateless protocol. what is the problem if http comes as statefull protocol?

- (A) If HTTP is statefull protocol for multiple requests given by client to web application single connection will be used between browser window and web server across the multiple requests. This may make clients to engage connections with web server for long time even though the connections are idle. Due to this the web server may reach to maximum connections even though most of its connections are idle.

To overcome that problem HTTP is given as stateless so no client can engage connection with webserver for long time. more ever the connection will be closed automatically at the end of each request related response generation. In internet environment since there is a chance of having huge amount of clients for each web site it is recommended to have stateless behaviour for http.

- 
- + If web application is capable of remembering a client data during a session across the multiple requests then that web application is called as stateful web application.
  - + In stateful web applications the web resource programs can use the data of previous request while processing current request that means while processing

request2 it can use request1 data and etc...

\* Even though http is stateless protocol we need to make our web applications as stateful web applications. For this we need to work with the following session tracking or session management techniques.

(1) Hidden Form Fields

(2) Http Cookies

(3) Http Session with Cookies

(4) Http Session with URL rewriting (recommended)

\* Session tracking / session management is all about making web application as stateful web application by remembering client data across the multiple requests during a session.

#### Real Time implementations of this session tracking / session management

(1) Remembering email id and password during gmail.com email operations

(2) Remembering username and password during online shopping operations in websites.

(3) Remembering previous forms data until last form data arrives in online applications / registrations

(4) while customizing webpage look in website as required for end user.

(5) while rendering direct advertisements in websites.

and etc...

\* If advertisements are rendered based on the kind of content surfing / browsing done by end user then that advertisement is called as direct advertisement.

Ex: In google.com when you search for the results of seven wonders we will get advertisement related to tours and travels.

#### working with Hidden Form fields based session tracking technique

\* An invisible text box of the form page is called as hidden box.

\* When form page is submitted the form data goes to the hidden box value goes to web resource program along with the request as request parameter value.

In form page

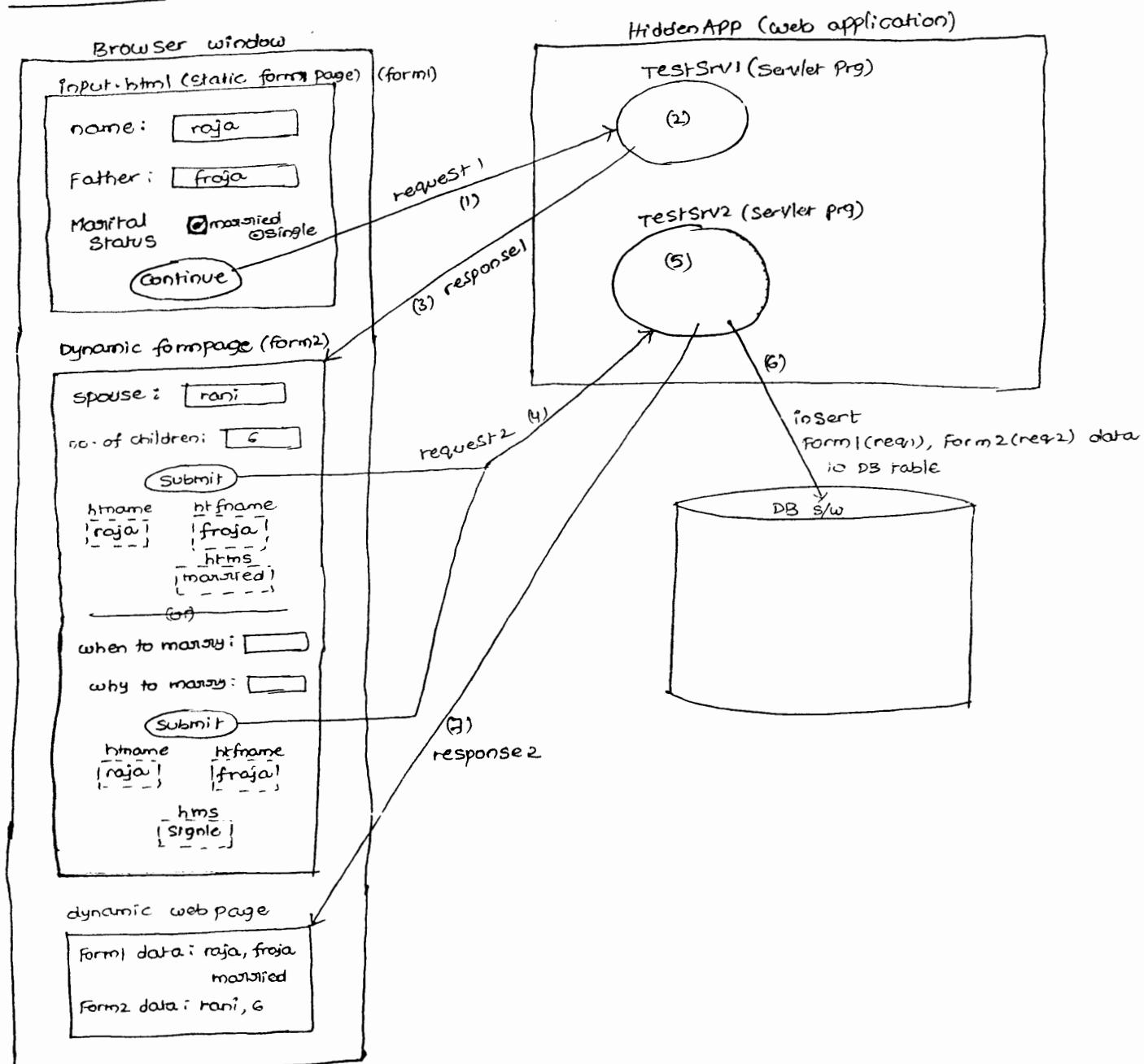
```
<input type="hidden" name="hi" value="hello"/>
```

In servlet program

```
String s1 = request.getParameter("t1");
```

gives "hello" to s1 variable

Making web application as stateful with the support of hidden boxes (hidden form fields)



\* In the above diagram TestSrv1 program receiving form1/request1 data and generating form2 dynamically. In this form2, form1 data is added as hidden box values so when TestSrv2 program gets request from form2 it is able to read form1/request1 data along with form2/request2 data. This is nothing but session tracking based on hidden form fields.

- \* The above diagram based web application is stateful because TestSrv2 is able to use request1 data while processing request2.
- \* For above diagram based application refer application(II) of the page no 76-78

### Advantages of Hidden form fields based session tracking technique

- \* Basic knowledge of HTML is enough to work with this technique.
- \* Hidden boxes resides in web pages of browser window so they do not provide burden to the server.
- \* This technique can be used along with all kinds of server side technologies and all kinds of web servers and application servers.

### Disadvantages

- \* Hidden form fields can not store java objects as values. They can just store text values.
- \* Hidden boxes doesn't provide data secrecy because their data can be viewed through view source option.
- \* Hidden boxes data travels over the network along with request and response. So we can say network traffic will be increased.
- \* While creating each dynamic form page we need to add the previous form pages data as hidden box values. This increases burden on the programmer.

### Http Cookies

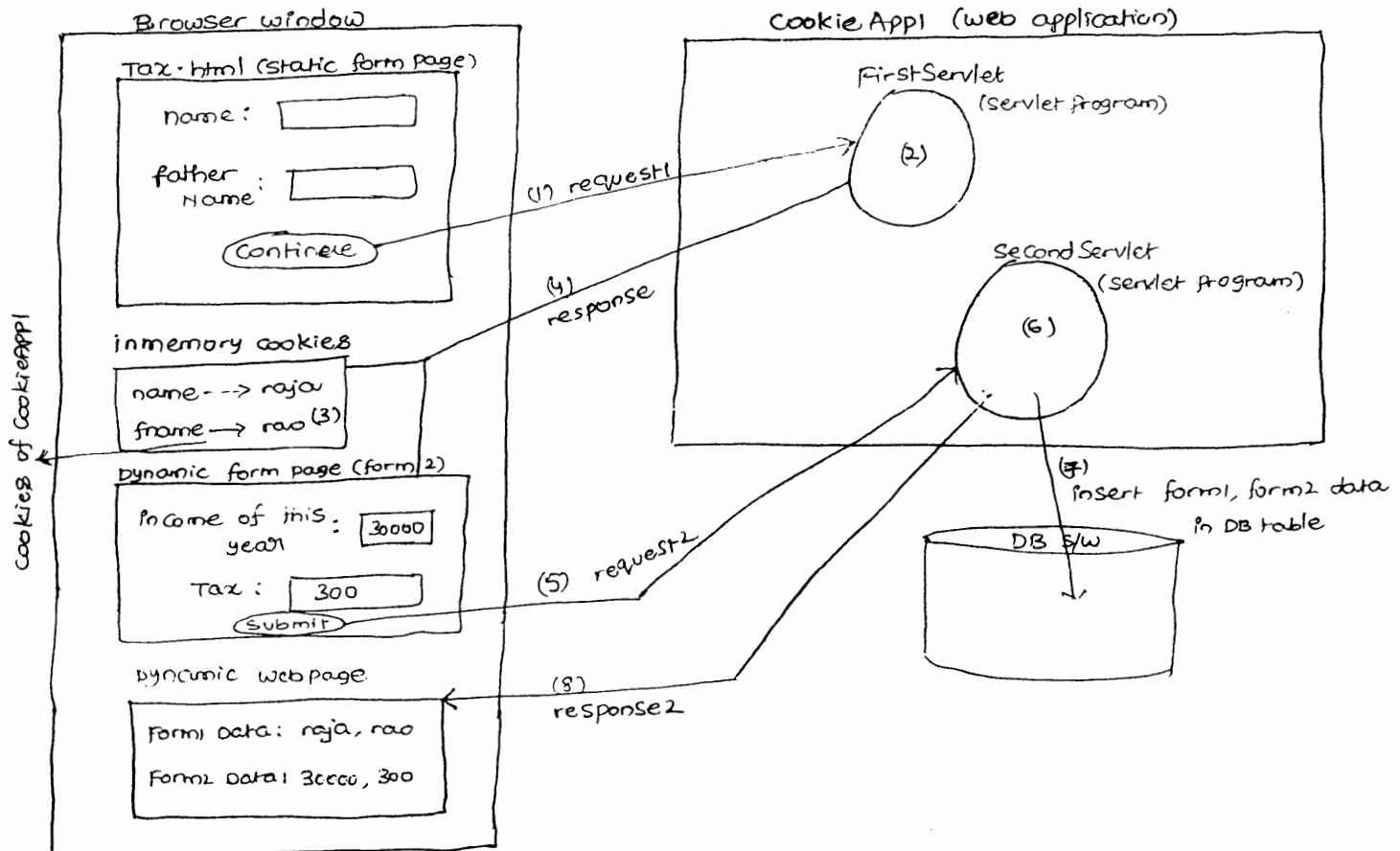
- \* Cookies are the small textual informations which allocate memory at client side by remembering client data across the multiple request during a session.
- \* The web resource programs of web application create cookies at server side but these cookies comes to client side along with the response and allocates memory at client side.
- \* Cookies of client side goes back to their web application along with the request given by clients (browser windows to web resource programs).
- \* There are two types of cookies
  - (1) Inmemory Cookies / per session Cookies
  - (2) persistent Cookies

(1) → allocates memory in browser window having "-1" as the expiry time (no expiry time). These cookies will be destroyed automatically once browser window is closed.

(2) → These cookies allocate memory in the files of client machine hard disk having positive number as expiry time. These cookies will not be destroyed when browser window is closed but they will be destroyed when their expiry time is completed/reached.

\* The cookie that is created without expiry time is called as in memory cookie. The cookie that is created with the expiry time is called as persistent cookie.

\* To work with cookies we must deal with pre defined http Servlet class based server programs.



\* Every cookie contains name and value as string information.

\* At client side or browser window we can see multiple cookies belonging to different domains or web applications. Every cookie remembers the web application name or domain name for which it is created.

\* Cookies of one web application that are there at client side will go to their web application along with the request then browser window gives request back to that web application.

With respect to the diagram

- (1) The form page Tax.html gives request1 to FirstServlet program of CookieApp1 web application.
  - (2) FirstServlet program reads form1 data, creates two in memory cookies having that form1 data.
  - (3), For(4) → FirstServlet program generates response to browser window adding the cookies & having dynamic form page. These cookies allocate memory on browser window as in memory cookies.
  - (5) The dynamic form page (form2) generates request to second servlet program of CookieApp1 application so the two cookies of CookieApp1 will go to SecondServlet program along with the request.
  - (6) SecondServlet program reads form2 data normally but reads form1/request1 data from the cookies of request2. This indicates SecondServlet is able to use form1/request1 data while processing form2 request/request2. (which is nothing but session tracking).
  - (7) SecondServlet writes both form1, form2 data to db table.
  - (8) SecondServlet program generates dynamic web page having form1 and form2 data.
- \* The web resource programs of web application send cookies to client machine along with the response as 'set-cookie' response header value.
- \* The browser window sends the cookies of back to web application along with the request given to web resource programs as 'cookie' request header value.

Cookie API : (working with javax.servlet.http.Cookie class)

to create cookies and to add them to response

```
Cookie ck1 = new Cookie("ap", "hyol"); //creating new cookie  
          ↓      ↓  
          cookie   cookie  
          name    value  
          (must be string)
```

```
resp.addCookie(ck1); //adding cookie to response
```

↓  
Here ck1 cookie is in memory cookie (because no expiry time is set)

```
Cookie ck2 = new Cookie ("mh", "mumbai");
```

```
ck2.setMaxAge(1800); → setting expiry time for cookie
```

```
resp.addCookie(ck2); → adding cookie to response
```

Here ck2 cookie is persistent cookie having 1800 seconds as expiry time.

to know max age (Expiry time) of cookie

```
int time = ck1.getMaxAge(); → gives -1 (for in memory cookie)
```

```
int time = ck2.getMaxAge(); → give 1800 seconds
```

to modify cookie value

```
ck1.setValue("new hyd");
```

```
ck2.setValue("navi mumbai");
```

To know domain name / web application name of the cookie

```
String d1 = ck1.getDomain();
```

```
String d2 = ck2.getDomain();
```

to set comment to cookie

```
ck1.setComment("holds ap's capital city");
```

```
ck2.setComment("holds mh's capital city");
```

to get comment of cookie

```
String c1 = ck1.getComment();
```

```
String c2 = ck2.getComment();
```

to read cookie values

```
String ck[] = req.getCookies();
```

↓  
reads all the cookies from the current http request.

```
if (ck != null)
```

```
{
```

```
for (int i=0; i<ck.length; i++)
```

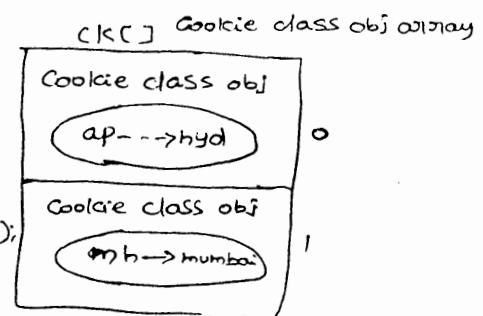
```
{
```

```
pw.println("Cookie name" + ck[i].getName());
```

```
pw.println("Cookie value" + ck[i].getValue());
```

```
}
```

```
}
```



To delete Cookies

- \* Cookies can not be deleted manually but programmatically.
- \* In memory Cookies will be destroyed automatically once their browser window is closed. whereas as persistent cookies will be destroyed if their expiry time is reached / completed.
- \* for example application to know the basics and behaviour of Cookies refer application (12) of the page nos (78) and (79).
- \* In windows xp while working with internet explorer the persistent cookies of web application will be stored in the files created in c:\documents and settings\administrator\cookies folder and the notation of the file is:

<windows user> @ <web application name> [<cookies count>].txt

Ex: administrator @ cookieApp [2].txt

writing HTML code in servlet program with attribute values of html tags

```
pw.println("<table border='1' align='center'>");  
          (or)  
pw.println(" <table border='1' align='center'>");  
          (or)  
pw.println("<table border='1' align='center'>");
```

- \* If we try to modify the content of the files where persistent cookies are stored then persistent cookies including that file will be destroyed.
- \* Internet explorer allocates the memory for in memory cookies on browser window whereas the memory for persistent cookies will be allocated as files on the client machine hard disk.
- \* In netscape navigator both in memory and persistent cookies allocate memory on browser window but persistent cookies will be there with expiry time.

#### NOTE:

- \* Netscape navigator also internally uses client machine hard disk files for storing persistent cookies.
- \* for source code of example application based on the above diagram refer application (13) (13) of the booklet page numbers (79) to (81)

(79) to (81)

- \* Every cookie remembers the domain name or web application name to which it belongs to.
- \* Cookies belonging to yahoo.com website will go to yahoo.com website only when browser window or client gives request back to that yahoo.com website.
- \* In yahoo-mail.com, gmail.com website related login page  remember me on this computer option stores the given login details (username, password) as persistent cookies on client machine hard disk. That means by using persistent cookies we can remember client data during session and after session.

### Advantages of Http Cookies :

- (1) Cookies allocate memory at client side so they do not give any burden to server.
- (2) All server side technologies and all web servers, application servers support cookies.
- (3) Persistent cookies can remember client data during session and after session with expiry time.

### Disadvantages

- (1) Cookies can not store java objects as values. They can store only strings.
- (2) Cookies data can be viewed <sup>through</sup> browser setting or through the files of file system. So they do not provide data security.
- (3) Cookies can be deleted explicitly through browser settings it may fail session tracking.
- (4) There is a restriction on no. of cookies that can be there in browser window. Per browser window and per web application max of 20 cookies. All together max of 300 cookies per browser window.
- (5) Cookies can be blocked/restricted coming to browser window. This fails session tracking.

I E → to delete cookies

Tools → Internet options → delete cookies

Netscape to delete

Tools → Cookie manager → remove all cookies

IE → to block cookies

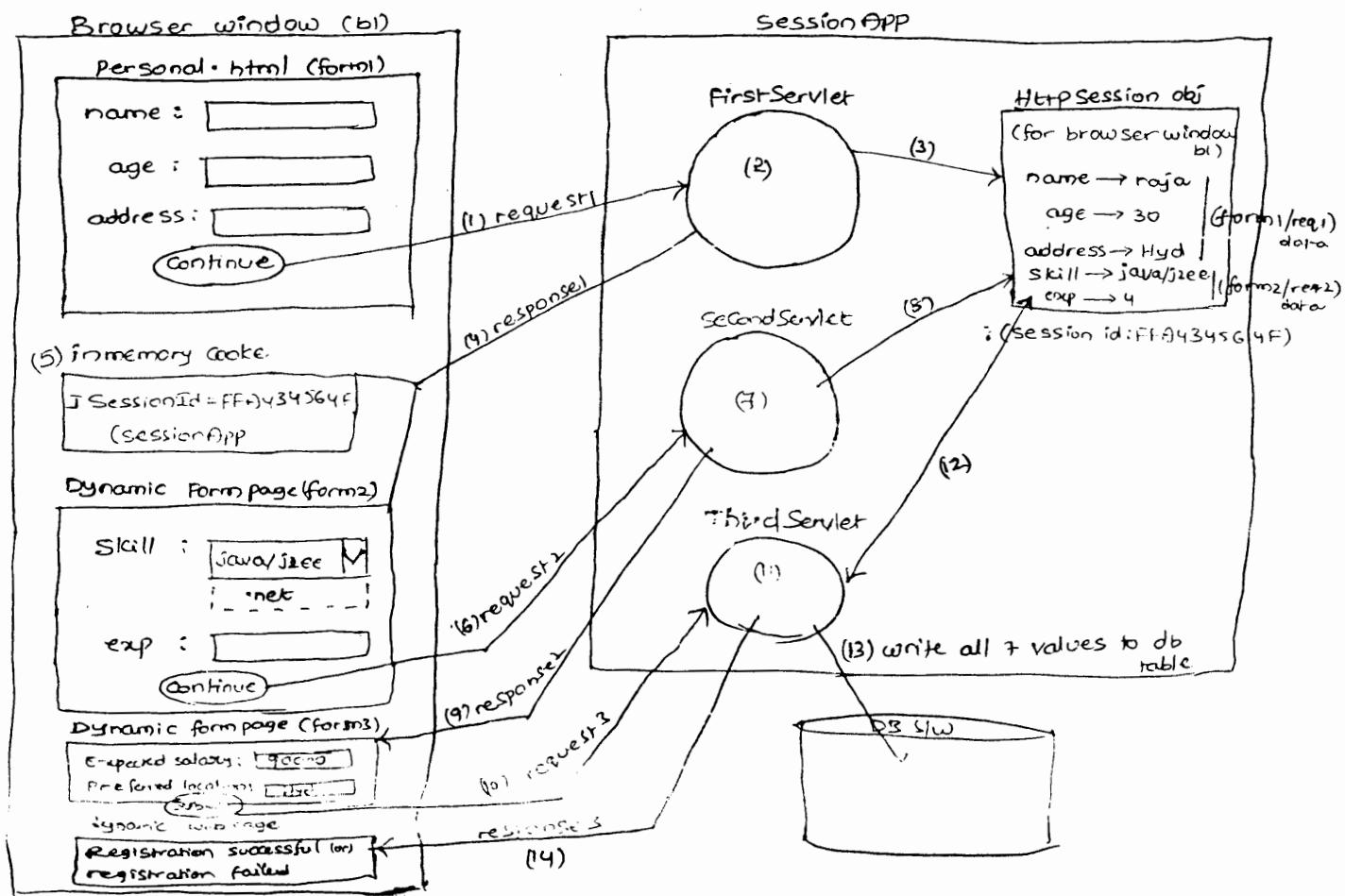
Tools → Internet Options → Privacy → use slider

Netscape → to block cookies

Tools → Cookie manager → Block Cookies from this site

### Http Session with cookies :

- \* HttpSession object allocates memory on the server and remembers client data across the multiple requests in the form of session attribute values.
- \* HttpSession object is one per browser window (client) so each HttpSession object can be used to remember client data during a session.
- \* HttpSession object means it is the object of a servlet container supplied java class implementing javax.servlet. http. HttpSession interface.
- \* Every HttpSession object contains session id and this session id goes to browser window from web application and comes back to browser web application from browser window in the form of in memory cookie value so this technique is called HttpSession with cookies technique.



- with respect to the diagram
- (1) Browser window b1 launches personal.html (form1) and sends the request to FirstServlet of sessionApp application.
- (2) FirstServlet reads Form1 data / request1 data.
- (3) FirstServlet creates HttpSession object on the server for browser window b1 and writes Form1 / request1 data to that session object as session attribute values.
- (4) FirstServlet generates form2 as dynamic form page and sends the session id of HttpSession object to browser window as in memory cookie value.
- (5) The in memory cookie having session id of HttpSession object (b1) allocates memory on the browser window. That cookie also remembers its web application name sessionApp.
- (6) End user fills up the form page (form2) and sends the request2 to Second Servlet of sessionApp. Along with the request the session id will go as cookie value.
- (7) Second Servlet reads form2 data.
- (8) Second Servlet uses the session id read from the cookie to get access to the HttpSession object of browser window b1 and writes form2<sup>request2</sup> data to that HttpSession object as session attribute values.
- (9) Second Servlet generates form3 as dynamic form page.
- (10) form3 generates request3 to third servlet of sessionApp application. Along with this request session id goes to third servlet as cookie value.
- (11) Third Servlet reads form3 data and session id from the cookie.
- (12) Third Servlet uses session id collected from the cookie to get access to HttpSession object of browser window b1 and reads form1 / request1, form2<sup>request2</sup> data from the session attributes of that HttpSession attributes. This indicates ThirdServlet is able to use request1 / form1 data and request2 / form2 data while processing form3 request or request3. This is nothing but session tracking.
- (13) Third servlet writes form1, form2, form3 values to the db table as a record.
- (14) Third Servlet generates dynamic web page having confirmation message.

- \* When HttpSession object is created the session id of that session object automatically goes to browser window as in memory cookie value.
- \* In HttpSession with Cookies session tracking technique based web application the browser window (client) will be identified across the multiple request during a session based on the session id sent by the browser window along with the request.

### Session API (working with javax.servlet.http.HttpSession Interface)

#### To create/ locate HttpSession object

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

- \* This method creates new HttpSession object on server for browser window if HttpSession object is not already available for browser window otherwise this method provides access to <sup>existing</sup> HttpSession object of that browser window.
- \* This method can create the new session between browser window and web application if session is not already between them otherwise this method makes current request to join in the existing session.

- \* If this method is called in FirstServlet program of above diagram then it makes request1 of browser window b1 begining new session between browser window b1 and web application session APP. If same method is called SecondServlet, ThirdServlet then it makes request2, requests of browser window b1 participating in existing session.

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

- \* This method ~~creates~~ gives access to existing HttpSession object of browser window, if not available this method returns null (indicating new HttpSession object can not be created).
- \* When this method is called the request can always join in existing session but can not create new session between browser window and web application.
- \* In the above diagram it is recommended to call `req.getSession()` in FirstServlet and to start the session and it is recommended to call `req.getSession(false)` method in SecondServlet, ThirdServlet to make their participating in existing session.

c) HttpSession ses = req.getSession(true);  
    "same as (a)"

Conclusion:

to create new session/to locate existing session → use (a), (c) options

only to locate existing session → use (b) option.

To know Session ID

String id = ses.getId();

To know session object creation time / session started time

long ms = ses.getCreationTime();

Date d1 = new Date(ms);

\* In the above code ms represents milliseconds that are elapsed from  
1970 Jan 1st 00:00 hours to the date and time of HttpSession object creation.  
Epoch standard

\* To know the last accessed time of HttpSession object

long ms = ses.getLastAccessedTime();

Date d2 = new Date(ms);

\* The above code gives the last accessed date and time of HttpSession object.

To get access to ServletContext object

ServletContext sc = ses.getServletContext();

To know whether session is new or not

boolean b = ses.isNew();

\* This method returns true when ses object is new object and just created object and its session id still yet to be delivered (sent) to client (browser window) otherwise this method returns false. (when session object is old object) and its session id is already there with client).

\* If this method is called in FirstServlet program of diagram (which receives only first request of the browser window) then this method returns true. If this method is called in otherServlet programs of web application then this method returns false.

## To invalidate the session

\* Invalidating the session is nothing but closing the session between browser window and web application. In this process the entire data from session object will be removed and makes session object inactive object, ready for garbage collection.

a) when ses.invalidate() method is called

b) when browser window is closed

NOTE: since session id will be stored as in memory cookie value of browser window and that in memory cookie will be destroyed once browser window is closed so the session will be invalidated once browser window is closed.

c) when MaxInactiveInterval / SessionIdleTimeout period is completed/reached

\* If session object is continuously idle for certain amount of time then it will be invalidated automatically. The default session idle timeout period is 30 minutes (in most of the servers). But this can be changed explicitly either by using programmatic approach or declarative approach.

### (I) Programmatic approach (Java code)

In server prg/ JSP prg

ses.setMaxInactiveInterval(1500);

### (II) Declarative approach (xml code)

In web.xml

<web-app>

<session-config>

<session-timeout>20 </session-timeout>

</session-config>

↓  
minutes

</web-app>

\* once session idle timeout period or maxInactiveInterval period is completed the underlying web server automatically expires the session (invalidates the session).

### To know current maxInactiveInterval period / SessionIdleTimeout period

```
int t1 = ses.getMaxInactiveInterval();
```

NOTE: The above method returns 30 if no value explicitly set as sessionidle timeout period.

If you set sessionIdleTimeout period in both programmatic and declarative approaches with two different values, can you tell me which value will be effected finally?

(a) Since the code of servlet program executes after web.xml code so the time specified through programmatic approach will be effected by overriding the time specified through declarative approach.

\* In HttpSession object the client data will be preserved in the form of session attribute values.

\* HttpSession object and its session attributes are visible and accessible in all web resource programs of web application but they must get request from that particular browser window (client) for which this session attribute and session objects are created.

To create / modify session attribute

```
ses.getSession.setAttribute("age", new Integer(30));
```

ses.putValue(-,-) is deprecated method of setAttribute(-,-)

To read session attribute value

```
Integer s1 = (Integer)ses.getAttribute("age");
```

ses.getValue(-,-) is deprecated method of getAttribute(-,-)

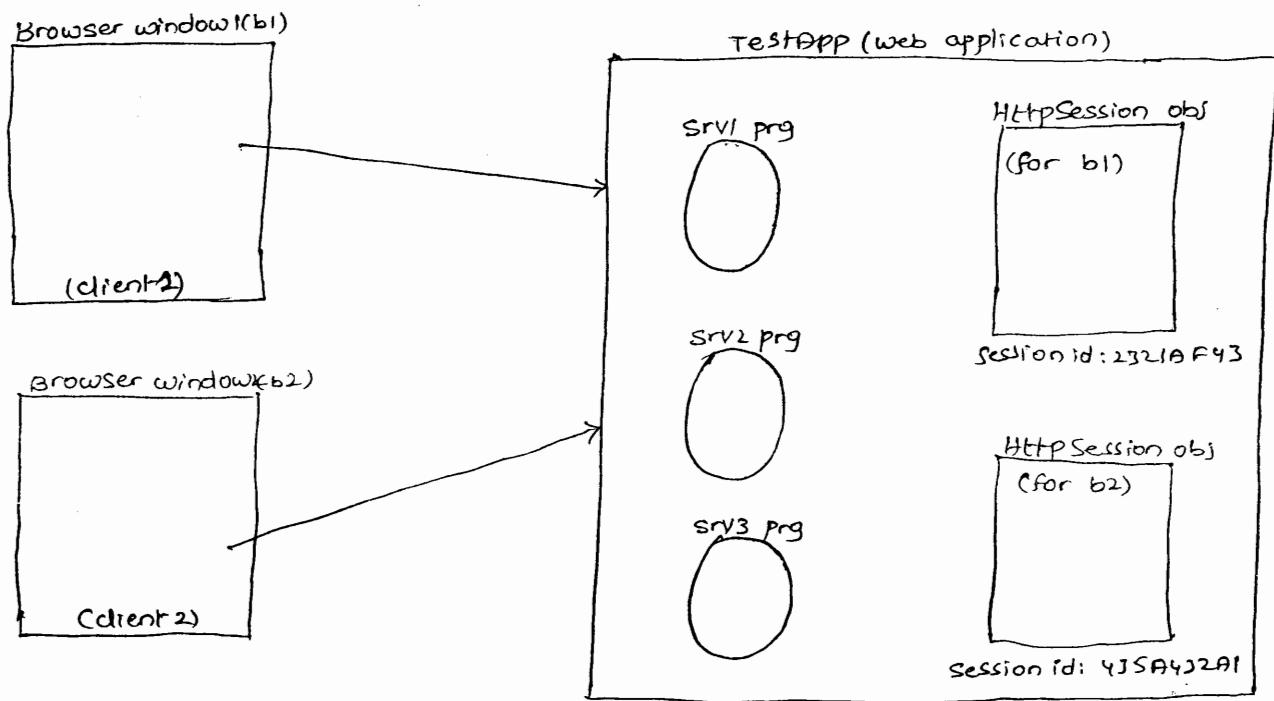
To remove session attribute

```
ses.removeAttribute("age");
```

ses.removeValue(-,-) is deprecated method of removeAttribute(-,-)

\* For example application on HttpSession with cookies based on <sup>previous</sup> diagram refer page nos 85 - 88 81 - 85

+ If multiple clients (browser windows) are giving request to a web application in which HttpSession object based session tracking is enabled then for every browser window one HttpSession object will be created on the server having unique session id.



- \* In HttpSession with cookies session tracking enabled web application the moment HttpSession object is created on server for browser window the servlet container automatically creates one inmemory cookie having session id and adds that cookie to the response to send to the browser window (no need of performing manual work).
- \* while working HttpSession object based web application if browser window is closed in the middle of the session then the existing session will be invalidated and that session will not be continued with new browser window.
- \* while working with HttpSession object based application what happens if underlaying server is restarted in the middle of the session ?

(i) The session will be continued.

Server collects the data of HttpSession objects and writes to files through serialization process having session ids when programmer shutdown the server. When server is restarted HttpSession object will be created having old data and old session ids by reading data from the above said file through de-serialization process. Due to this the session will be continued even though the server is restarted in the middle of session.

Advantages of HttpSession with Cookies session tracking technique:

i) HttpSession objects allocates memory on the server holding client data

during session as session attribute values. So there will be data security for client's session data.

(2) Client data during session will not travel along with request and response over the network so this reduces network traffic between client and web server.

(3) The session attributes of HttpSession objects can take Java objects as values.

(4) All Java based web servers and application servers support this technique.

(5) This technique allows the programmer to specify session idle timeout period to invalidate inactive session objects.

#### Disadvantages:

(1) HttpSession objects allocate memory on the server. This increases burden on the server.

(2) If cookies are restricted coming to browser window this technique fails to perform session tracking.

#### HttpSession with URL rewriting

\* The limitation with third technique is if cookies are restricted to coming to browser window then third technique fails to perform session tracking because it uses in-memory cookie to send session id to browser from web application and to bring session id back to web application from browser window along with request.

To overcome that problem work with HttpSession with URL rewriting which does not use cookies to send and receive session id. Moreover this technique appends session id to a url that goes to browser window from web application along with the response and that comes back to web application from browser window along with the request. Generally we append session id to the action url (the action attribute of form tag) of dynamic form page generation code.

#### NOTE:

res.encodeURL("s2url"); → gives s2url; ;sessionid: 4243541A3F723

This method uses URL appended with current object's session id as shown above.

\* By taking above kind of URL as action URL of dynamic form page we can perform http session with URL rewriting based session tracking.

#### In Servlet program

```
pw.println("<form action = " + res.encodeURL("surl") + " method = get>");  
pw.println("-----");  
pw.println("-----");  
pw.println("</form>");
```

\*\*\*  
↓  
keeps the url appended with sessionid as the  
action attribute value of <form> tag

\* For example application on HttpSession with URL rewriting refer application 15 of the page nos 85-88.

#### Advantages and disadvantages of HttpSession with URL rewriting

same as HttpSession with Cookies technique but this technique also works even though cookies are restricted coming to browser window.

#### Conclusion on Session tracking technique

\* While developing large scale and commercial website which contains huge customer base take the support of Http Cookies technique.

Ex: www.gmail.com, www.yahoo-mail.com

\* While developing medium scale and certain organization based website which contains limited amount of customers use HttpSession with URL rewriting technique.

Ex: www.citibank.com, www.satyatechnology.it.com

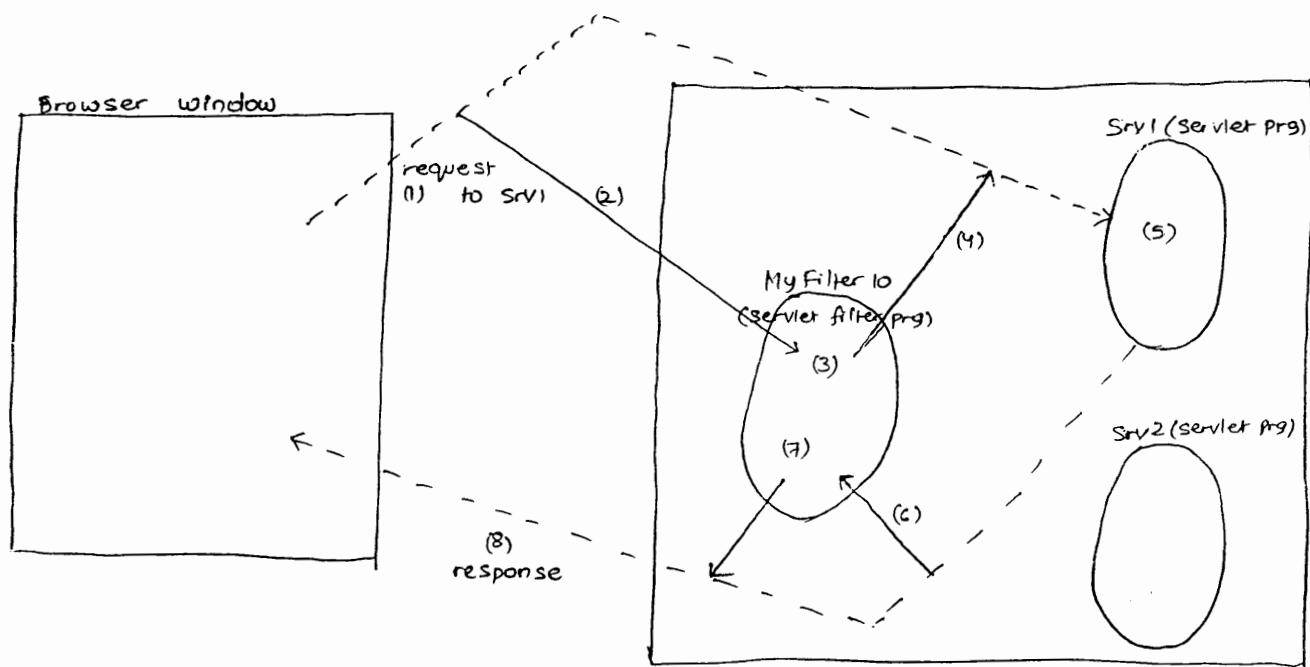
\* We can use multiple session tracking techniques in a single web application development. For example the online shopping websites like www.yebhi.com uses both Http Cookies, HttpSession with URL rewriting techniques.

↓  
for holding shopping cart  
information

↓  
for holding login details (username, password)

### Servlet filters :

- \* A servlet filter is a special web resource program of java web application that is capable of trapping and taking request and response of other web resource programs of that web applications.
- \* In Servlet filter program we can keep the common and global pre request processing logic and post response generation logic
- \* To add new functionalities to web application without changing the source code of existing web resource programs we can take the support of servlet filter programs.



With respect to the above diagram

- (1) Browser window generates request to SrV1 servlet program.
- (2) The servlet filter program traps and takes that request.
- (3) Servlet filter program executes the common and global pre requesting process logic (like authentication logic)
- (4) Servlet filter program forwards the request to actual SrV1 servlet program.
- (5) The main request processing logic of SrV1 program executes and generate response.
- (6) Servlet filter program traps and takes the response.
- (7) Filter program executes the common and global post response generation logic (like adding more content at the end of response).
- (8) Servlet filter program sends response to browser window.

## Servlet filter basics

- \* Servlet filter program is a java class that implements javax.servlet.Filter interface.
- \* Every servlet filter program must be configured in web.xml file using filters filter mapping tags.
- \* To link servlet filter program with servlet program of web application the url pattern of servlet program must be taken as the url pattern of servlet filter program.
- \* Servlet container creates our servlet filter class object either during server start up (or) during the deployment of the web application.
- \* We can map servlet filter program with one servlet program, multiple servlet programs (or) all servlet programs of web application.
- \* There are 3 lifecycle methods in servlet filter program
  - (1) public void init(FilterConfig fg)
  - (2) public void doFilter (servletRequest req, servletResponse res, FilterChain fc)
    - ↓
    - This object points to the target mapped Servlet programs of filter program.
  - (3) public void destroy()
- (1) → executes during instantiation and initialization process of our servlet filter program.
- (2) → executes for every request trapping and response trapping.
- (3) → executes when Servlet Container is about to destroy our servlet filter class object.

There are 3 types of filter programs.

- (1) request Filter (Contains only pre request processing logics)
- (2) response Filter (Contains only post response generation logics)
- (3) request - response Filter (Contains both pre-request processing and post response generation logics)

## Examples of request filters

Authentication Filter

Authorization Filter

## Examples of response filters

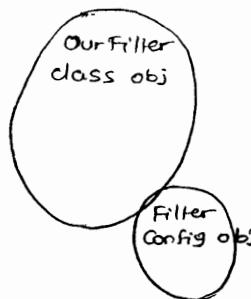
Signature Filter

Auges Filter → it removes html tags from response of the servlet programs which are not supported by browser window).

compression filter (reduces the size of the response content)

## Examples of request-response Filter

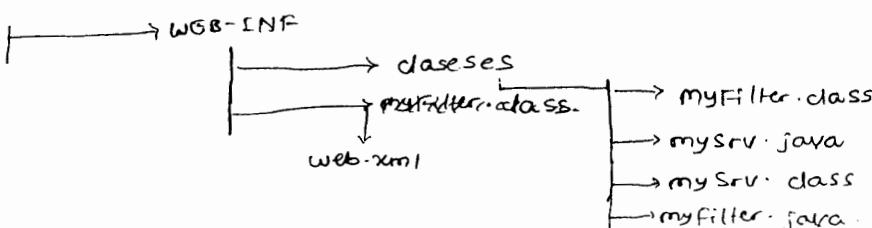
performanceTestFilter (This filter holds request trapping time and response trapping time of certain Servlet program and calculates difference between both timings to decide the performance of the Servlet program).



- \* FilterConfig object is right hand object of our servlet filter class object and it can be used to read init parameter values of filter program in web.xml file.
  - \* init( ) method of Servlet Filter program contains initialization logic.
  - \* doFilter(---) of Servlet Filter program contains pre-request processing logic and post response generation logics.
  - \* destroy( ) of ServletFilter program contains uninitialization logic.

understanding flow of execution for servlet filter programs when it is mapped with a servlet program.

①



## MyFilter.java

```
②  
public class MyFilter implements javax.servlet.Filter
{
    public void init(FilterConfig cg) { ③
        //initialization logic
    }

    public void doFilter(ServletRequest req, ServletResponse res, FilterChain fc) { ④
        // ⑤ } Pre-request processing logic
        fc.doFilter(req, res); ⑩
        // ⑪ } Post-response generation logic
    }
}
```

```
public void destroy()
{
    ===== //uninitialization logic
}
```

### MySrv.java (servlet program)

```
public class MySrv extends HttpServlet
{
    public void service(--) throws SE, IOE
    {
        ===== (15)
    }
}
```

### web.xml

```
<Servlet>          (13)
    <Servlet-name> abc </Servlet-name>
    <Servlet-class> MySrv </Servlet-class>
</Servlet>          (14)

<Servlet-mapping>   (12)
    <Servlet-name> abc </Servlet-name>
    <url-pattern> /test1 </url-pattern>
</Servlet-mapping>   (11)

<filter>            (1)
    <filter-name> xyz </filter-name>
    <filter-class> MyFilter </filter-class>
</filter>            (8)

<filter-mapping>    (6)
    <filter-name> xyz </filter-name>
    <url-pattern> /test1 </url-pattern>
</filter-mapping>    (5)
                                ↓
                                matching with servlet program (mySrv) url pattern
</web-app>.
```

### Browser window

http://localhost:2020/TestApp/test1 (4)

(1) response will be displayed on browser window.

With respect to

- (1) programmer deploys TestApp web application in web server/application server
- (2) based on web.xml entries servlet container creates our filter class object (xyzFilter) either during server startup or during the deployment of the web application.

- (3) (3) servlet container completes initialization process on filter by executing the logic of init(-) method.
- (4) End user gives request to servlet program mySrv from browser window (having /test1 in the request url).
- (5), (6), (7), (8) → since MyFilter program is configured with MySrv program the servlet filter traps and takes the above request that is given to mySrv program. In this process servlet container locates MyFilter class object.
- (9) servlet container calls life cycle method doFilter(---) and executes pre-request processing logic.
- (10) The fc·doFilter(--) method call forwards the request to the target web resource program (mySrv).
- (11), (12), (13), (14) → servlet container completes instantiation and initialization of MySrv program based on its configuration done in web.xml file.
- (15) servlet container calls service method of MySrv program to process the request and to generate the response.
- (16) The filter program (MyFilter) traps the response of mySrv program (servlet program) and executes post response generation logic that is placed after fc·doFilter(--) method.
- (17) The filter program MyFilter sends the response to browser window.

#### sample web application

srv1, srv2, srv3 are servlet programs.

F1 is servlet Filter program.

to configure filter with all the servlet programs of web application

Configure F1 filter having url pattern /\*

to configure F1 filter with srv1, srv2 programs of web application

srv1 program url pattern : /x/y/s1url

srv2 program url pattern : /x/y/s2url

F1 filter program url pattern : /x/y/\*

to configure F1 filter with srv1 program of web application

srv1 program url pattern : /s1url

F1 filter program url pattern : /s1url

what is the difference between doFilter(---) of javax.servlet.Filter interface and doFilter(--) method of javax.servlet.FilterChain interface?

(A) doFilter(---) method of Filter interface is lifecycle method of ServletFilter program so programmer uses this method to place prerequest processing and post response generation logics.

doFilter(--) method of FilterChain is not lifecycle method so programmer uses this method to invoke next filter in the chain or to invoke the mapped servlet program or JSP Program of current servlet filter programs.

—. —

### sample web application

srv1, srv2, srv3 are servlet programs

F1, F2 are servlet filter programs

To configure F1, F2 filter programs with srv1 program

url pattern of srv1 program : /s1url

url pattern of F1 filter program : /s1url

url pattern of F2 filter program : /s1url

To configure F1, F2 filter programs with srv1, srv2 programs

url pattern of srv1 program : /x/y/s1url

url pattern of srv2 program : /x/y/s2url

url pattern of F1 filter program : /x/y/\*

url pattern of F2 filter program : /x/y/\*

To configure F1, F2 filter programs with all servlet programs of webapplication

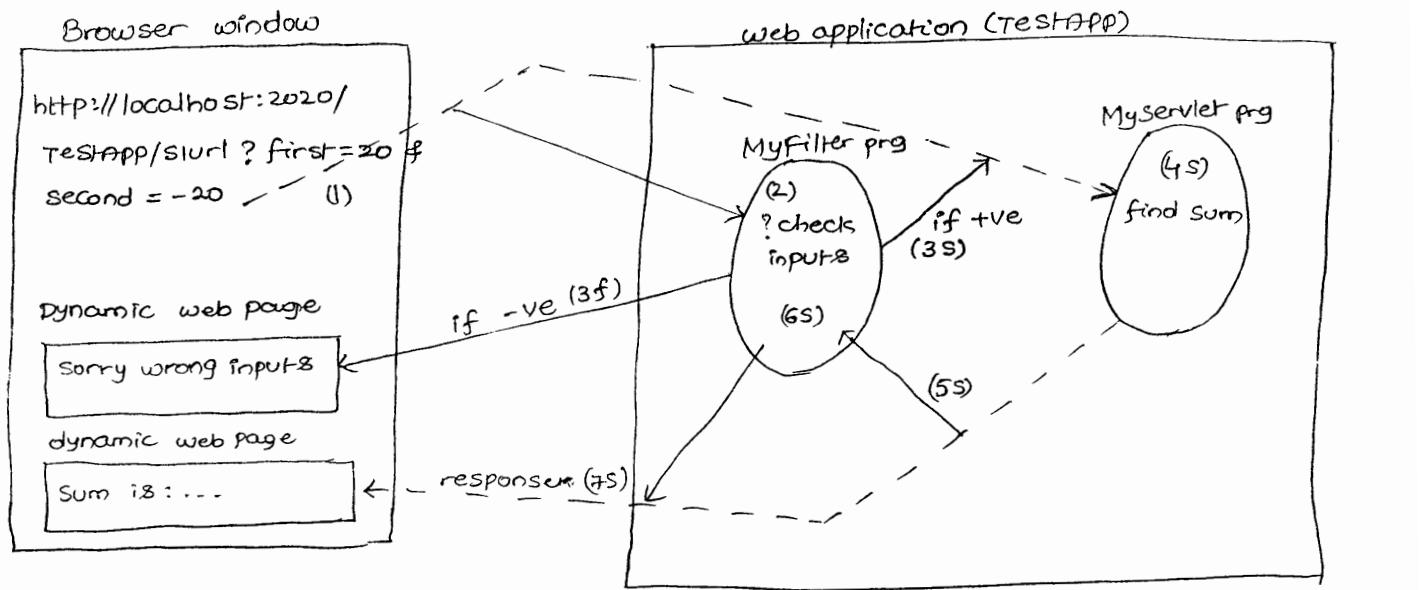
url pattern of srv1 F1 filter program : /\*

url pattern of F2 filter program : /\*

\* when filter is configured with servlet program then filter can trap the request and response of servlet program.

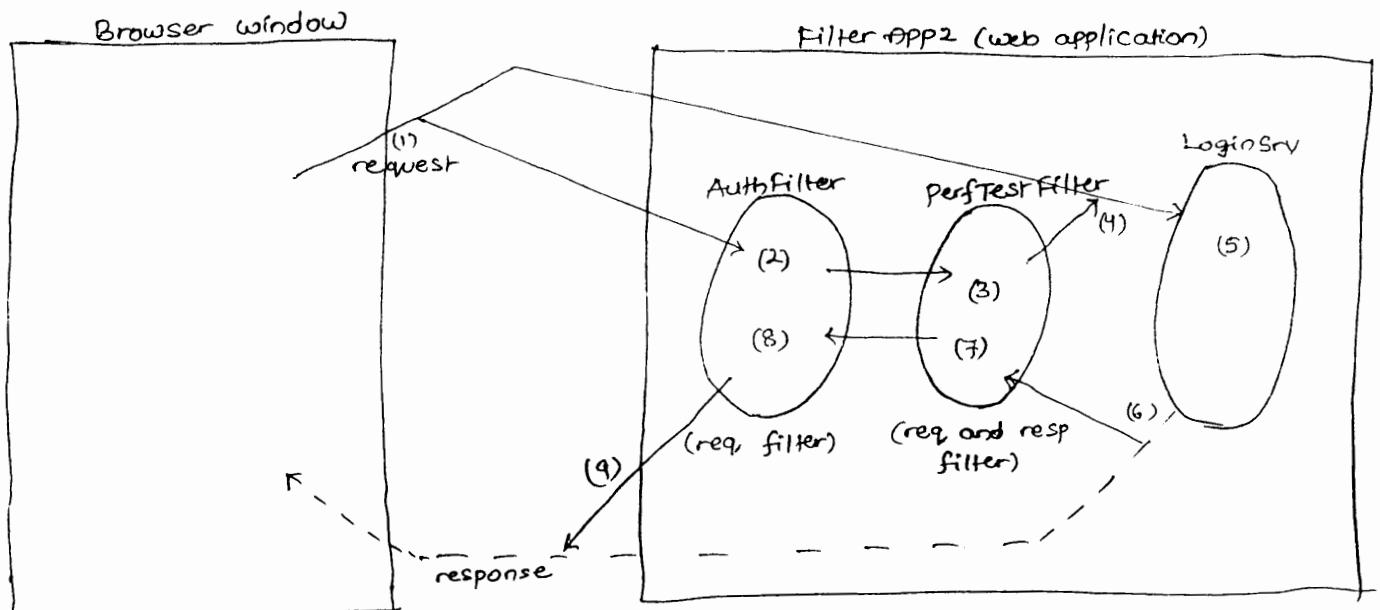
### Example application

\* In the below diagram MyFilter program traps the request going to myServlet program and checks the input values (request parameter values). If those values are positive then it forwards the request to MyServlet program (3S) otherwise it blocks the request in filter program and sends error response to browser window (3f)



\* In the above diagram MyFilter also traps the response of MyServlet program but does not contain post response generation logic so the above filter is called as request filter.

\* For above diagram based application refer application (16) of Page nos (89&90)  
Example application on filter



\* After taking request from client the amount of time taken by servlet program to process the request is technically called as response time of servlet.

- \* The tomcat server maintains log files in Tomcat-home\logs folder on one per day basis having date in the file name.
- \* Instead of using System.out.println() statements which writes log messages to server console it is recommended to use log() method of ServletContext object which writes the log messages to current day's log file of Tomcat-home\log folder.

```
ServletContext sc = getServletContext();
sc.log("Hello");
```

- \* JDBC code to perform username and password based authentication

```
PreparedStatement ps = con.prepareStatement("select * from userlist where
                                         username=? and password=?");
ps.setString(1, "raja");
ps.setString(2, "hyd");
ResultSet rs = ps.executeQuery();
if (rs.next())
{
    === //Authentication successful
}
else
{
    === //Authentication failed
}
```

Userlist (db table)

<u>username</u>	<u>password</u>
raja	hyd
ravi	Vizag
ramesh	Delhi

- \* For above diagram based web application's source code refer application(17) of the page nos 90-93.
- \* The process of checking the identity of a user through his username and password is called as authentication.
- \* The process of verifying the access permissions of a user on particular resource is called as authorization.

To count no. of requests (hits) coming to web application when web application is in active mode we can develop filter program as shown below.

Step(1): Develop the servlet filter program as shown below in WEB-INF\classes folder of web application (like URLAPP) having the logic of counting request.

```
//Countfilter.java
import javax.servlet.*;
import javax.servlet.http.*;
public class CountFilter implements Filter
{
    FilterConfig fg;
    public void init(FilterConfig fg)
    {
        this.fg=fg;
        int cnt=0;
    }
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain fc)
        throws ServletException, IOException
    {
        cnt++;
        ServletContext sc=fg.getServletContext();
        sc.setAttribute("hitcnt",cnt);
        fc.doFilter(req,res);
    }
    public void destroy()
    {
    }
}
```

Step(2): Configure above filter in web.xml file with url pattern /\*

Step(3): Read and display servletContext attribute value in every web resource program of web application.

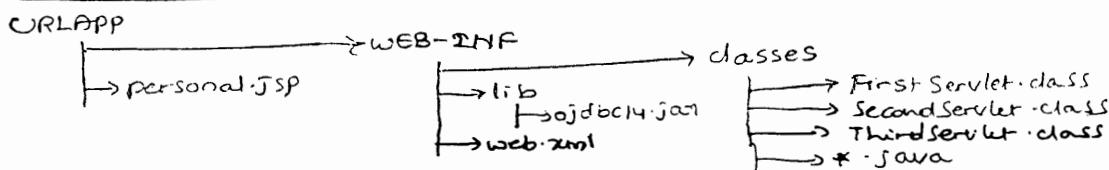
In personal.jsp

```
<% out.println("Request Count:" + application.getAttribute("hitcnt")); %>
```

In FirstServlet.java, SecondServlet.java, ThirdServlet.java

```
ServletContext sc= getServletContext();
pw.println("<br>Request Count" + sc.getAttribute("hitcnt"));
```

Deployment directory structure



## Request url

http://localhost:2020/URLAPP/personal.jsp

\* use application (s) as base application to develop this application.

## Applet to servlet communication

\* In web applications we need form pages to submit the request to web resource programs by having end user supplied data. These form pages provide Graphical User Interface to end user to submit the request to web resource programs having data. There are two ways to develop form pages.

- 1) As html form page (using <form>, <input> and etc... tags)
- 2) As java applet based form page.

\* there are two types of applets

(a) Trusted applet

(b) untrusted applet

\* An applet is a compiled Java class that can be sent over the network as web page.

\* When untrusted applet is launched in the browser window by gathering it from network it can't interact with the files of file system so untrusted applets can not write virus to the computer (that means provides security).

\* Trusted applets that are launched in the browser window can interact with file system and can write virus to file system (that means no security).

\* While designing form page if performance is important (should be launched very fast) then use html form pages. If security is important that take untrusted applet as form page.

## What are the differences between Applet and servlet?

### Applet

\* Client side technology to develop client side web resource programs.

\* Comes to browser window for execution.

\* Generate static form pages.

\* Need not be configured in web.xml file.

### Servlet

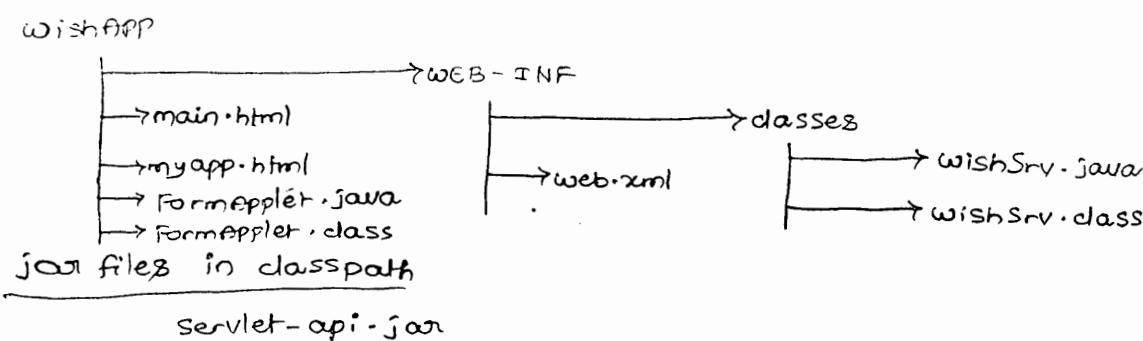
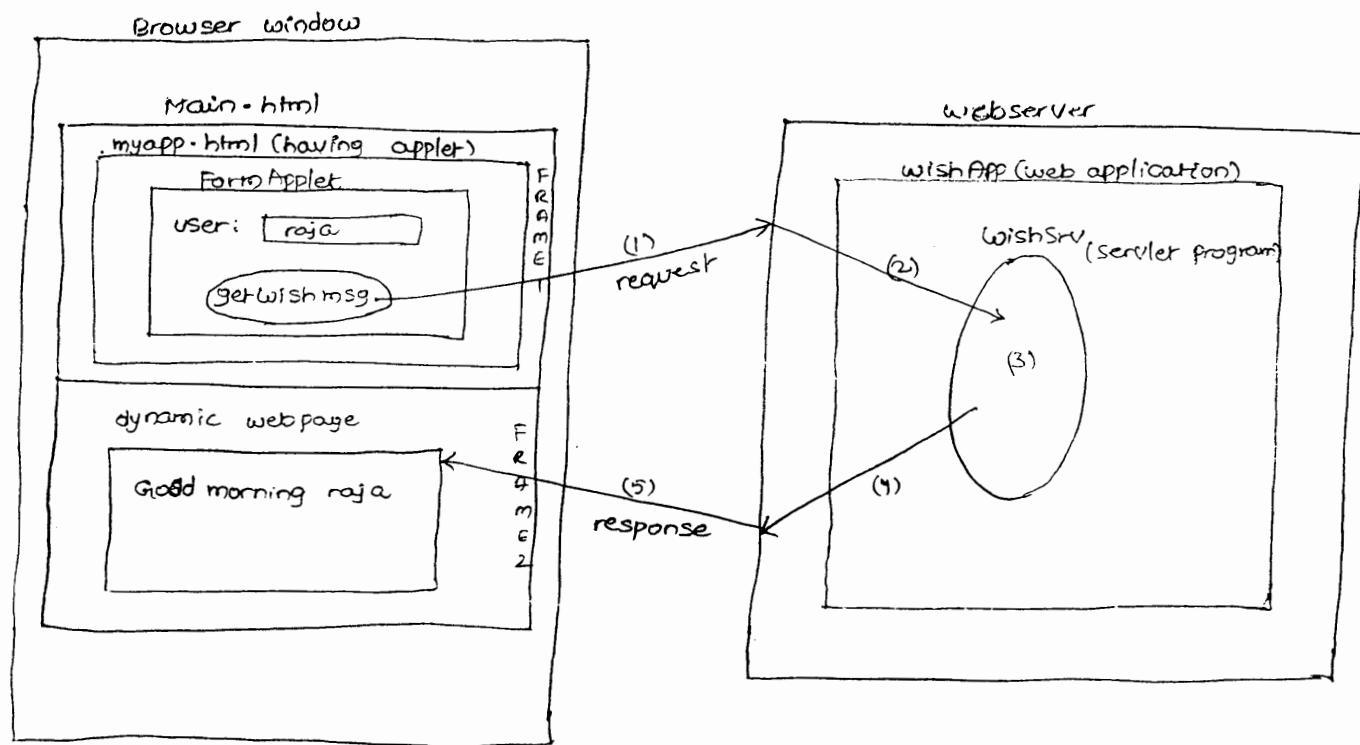
\* Server side technology to develop server side web resource programs.

\* Uses servlet container of web server for execution.

\* Generate dynamic web pages & form pages.

\* Must be configured in web.xml file.

- \* Uses life cycle methods for execution. They are `init()`, `start()`, `stop()`, `destroy()`, `paint()`
- \* Executes through life cycle methods. They are `init()`, `startService(-)`, `destroy()`.
- \* Applet to Servlet Communication means making the applet based form page sending request to servlet program having data.
- \* In html form page to servlet communication the html tags automatically prepares request URL by having form data as query string values whereas in applet to servlet communication all these operations should be performed by the programmer manually



### wishSrv.java

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

public class wishSrv extends HttpServlet
{
    public void service(HttpServletRequest req, HttpServletResponse res) throws
        ServletException, IOException
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        //read form data (applet data)
        String uname = req.getParameter("user");
        //write b.logic
        Calendar d = Calendar.getInstance(); //gives current date and time.
        int h = d.get(Calendar.HOUR_OF_DAY); //gives current day hour of day
        if (h<12)
            pw.println("Good morning " + uname);
        else if (h<=16)
            pw.println("Good afternoon " + uname);
        else if (h<=20)
            pw.println("Good evening " + uname);
        else
            pw.println("Good night " + uname);
        //close the stream object
        pw.close();
    }
}
```

### main.html

```
<frameset rows="40%, *">
    <frame name="f1" src="myapp.html" />
    <frame name="f2" />
</frameset>
```

+ frame with name is called as named frame.

### myapp.html

```
<applet code="FormApplet.class" width="200" height="200">
</applet>
```

### FormApplet.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

public class FormApplet extends Applet implements ActionListener
{
    Label l1;
    TextField tf1;
    Button b1;
    public void init()
    {
        l1 = new Label("User:");
        add(l1);
        tf1 = new TextField(10);
        add(tf1);
        b1 = new Button("Get wish");
        b1.addActionListener(this);
        add(b1);
    }
    public void actionPerformed(ActionEvent ae)
    {
        try{
            //example req url : http://localhost:2020/wishapp/wurl?user=raja++rao
            String qrystr = "?user=" + tf1.getText().replace(' ', '+'); //gives ?user=
            String requrl = "http://localhost:2020/wishapp/wurl" + qrystr;           raja++rao
            //prepare req url as URL class obj.
            URL url = new URL(requrl);
            //get Access to AppletContext obj
            AppletContext ac = getAppletContext(); //public method of Applet class
            //send request to servlet program
            ac.showDocument(url, "f2"); //sends the req to wishsrv prg and
                                         //displays the generated webpage in
                                         //a frame window whose name is "f2"
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

### web.xml

```
<web-app>
    <servlet>
        <s-n> xyz </s-n>
        <s-c> wishSrv </s-c>
    </servlet>
```

```
<s-m>
<s-n> xyz </s-n>
<u-p> /wurl </u-p>
</s-m>
</web-app>
```

\* use following request url to test the application

http://localhost:2020/wisApp/main.html

\* AppletContext object represents the current executing environment of applet programming so it can be used to make current applet talking with another applet or another web resource program of web application. To get this AppletContext object we can use getAppletContext() predefined java.applet.Applet class

#### File downloading :

\* selecting file from the client machine file system and sending that file to server machine file system through network is called as file uploading and reverse is called as file downloading.

\* All the files of a computer managed by operating system together is called as file system of Computer.

\* Servlet API gives built-in support for file uploading. Since it is complex we generally prefer working with third party API called Java Zoom API perform this file uploading.

\* Java Zoom API comes in the form of JAR files and we can download them from www.javazoom.net website. The JAR files are

uploadbean.jar (main JAR file)

struts.jar  
cos.jar } dependent JAR files to uploadbean.jar

\* The classes and interfaces of uploadbean.jar files uses the classes and interfaces of struts.jar file, cos.jar file.

\* If Java Web Application uses third party API (other than SDK API's, JEE API's) then add 3rd party API related <sup>main</sup>JAR files to classpath and add the third party API related main and dependent JAR files to WEB-INF\lib folder of web application.

Ex: If servlet, JSP programs of java web application uses third party api called Java Zoom api then add the Java Zoom api related main jar file uploadbean.jar file to classpath. Similarly the Java Zoom api related main and dependent jar files uploadbean.jar, struts.jar, cos.jar to WEB-INF\lib folder of web application.

\* To select a file from client machine file system the form page of client machine web application should have file uploading component. For that we take <input> tag as shown below.

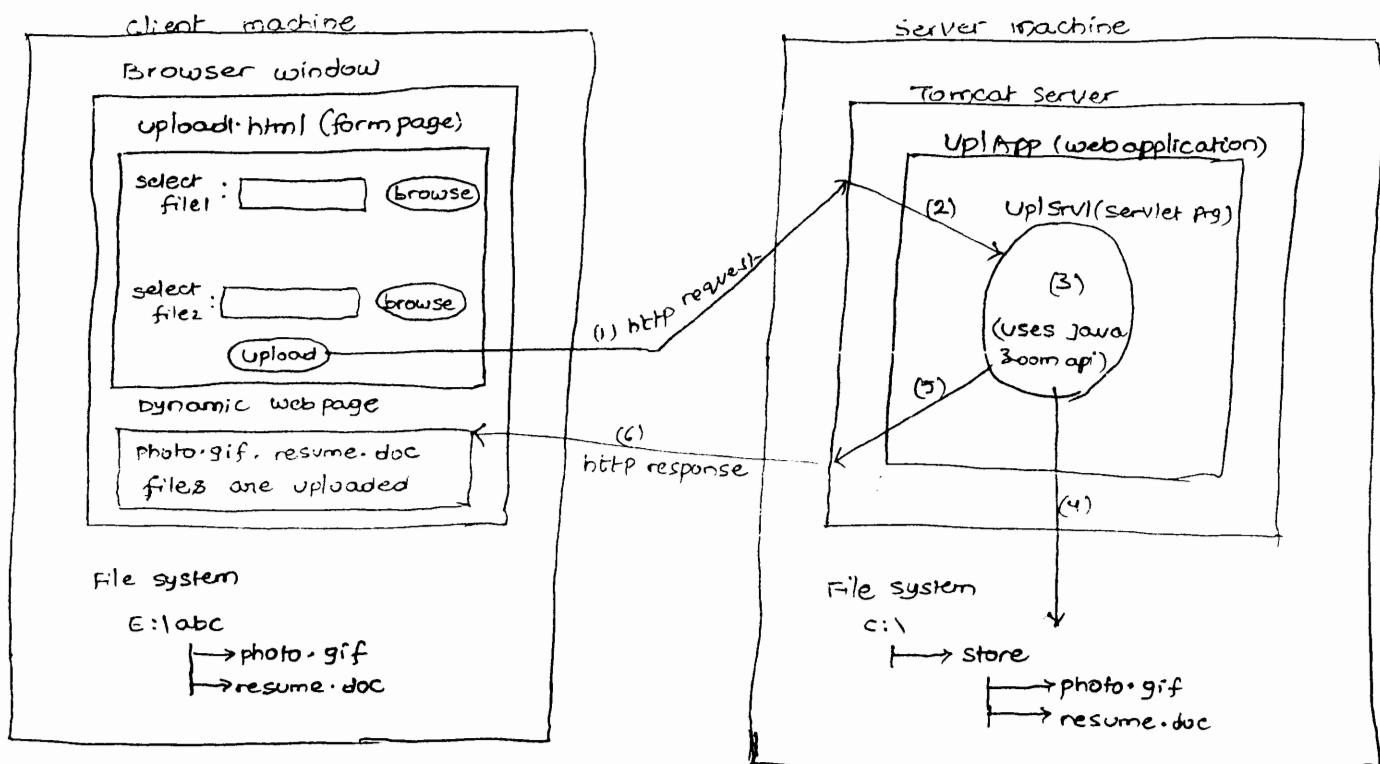
#### In form page

Select file: <input type="file" name="f1"/>

Select file:

file uploading component

\* The form page that contains file uploading component is recommended to generate "post" method based http request.



\* For the source code of above application refer application (18) that is given in page nos 93 and 94 of booklet.

\* we can see file uploading and downloading applications in email account applications (to send & receive attachments), job portal websites like naukari.com (job seeker uploads the resume, HR dept. of company downloads the resume) and matrimoni website

\* we can perform file downloading operations in two modes.

(1) making the response of web resource program as downloadable file

(2) Making download resources of the web application (img file, audio file, video file and etc...) as downloadable file through hyperlinks or buttons.

\* write the following two lines of content in web resource program like servlet, JSP to make that response of that web resource program as downloadable file.

```
res.addHeader ("Content-Disposition", "attachment; filename=abc.html");  
res.setContentType ("text/html");
```

↓  
response content type

↓  
response header

↓  
name of the downloadable file that holds response of the web resource program

\* An action performed on java object or component is called as an Event.

\* Executing some logics when event is raised is called as Event handling.

\* To perform event handling we use event listeners.

\* The reusable Java object is called as Component. we use AWT, swing concepts of JSE module to perform event handling.

\* To perform event handling we need four important details.

11) Source object on which the event will be raised (like button component)

(E) Event class name (like `java.awt.event.ActionEvent`)

3) Event Listener (like java.awt.event.ActionListener)

(iv) Event handling method (like public void actionPerformed(); )

+ From Servlet API 2.4 onwards we can perform event handling on request, session and ServletContext object through servlet listeners.

- Event handling on request object allows us to keep track of the creation and destruction of request object and we can use these timings to calculate request processing time of ~~eg~~ each request. This event handling also allows us to keep track of when each request attribute is created, modified and removed.

## Details that are required to perform event handling on request object.

Source object : request object

Event class : javax.servlet.ServletRequestEvent

Event Listener : java.servlet.ServletRequestListener

Event handling methods : (1) requestInitialized (-)  
(2) requestDestroyed (-)

} For Event handling on  
request object

Source object : request object

Event class : javax.servlet.ServletRequestAttributeEvent

Event Listener : java.servlet.ServletRequestAttributeListener

Event handling methods : (1) attributeAdded (-)  
(2) attributeRemoved (-)  
(3) attributeReplaced (-)

} For event handling on  
request attributes.

\* Event handling on ServletContext object helps us to keep track of when  
ServletContext object is created and destroyed. Based on this we can keep  
track of timings of web application deployment and undeployment or reloading  
or stop operations.

\* This Event handling can also be used to keep track of creation, modification,  
destruction of ServletContext attributes.

## Details that are required to perform event handling on ServletContext object

Source object : ServletContext object

Event class : javax.servlet.ServletContextEvent

Listener : javax.servlet.ServletContextListener (I)

Event handling methods : (1) ContextInitialized (-)  
(2) ContextDestroyed (-)

} For event handling on  
ServletContext object.

Source object : ServletContext object

Event class : java.servlet.ServletContextAttributeEvent

Listener : javax.servlet.ServletContextAttributeListener (I)

Methods : (1) attributeAdded (-)  
(2) attributeRemoved (-)  
(3) attributeReplaced (-)

} For event handling on  
ServletContext attributes

\* Event handling on session object can keep track of when HttpSession object  
is created and destroyed. Using this we can know how much time there the

User is there in the session. we can also use this event handling to keep track of when each session attribute is created or modified or destroyed.

Details that are required to perform event handling on HttpSession object

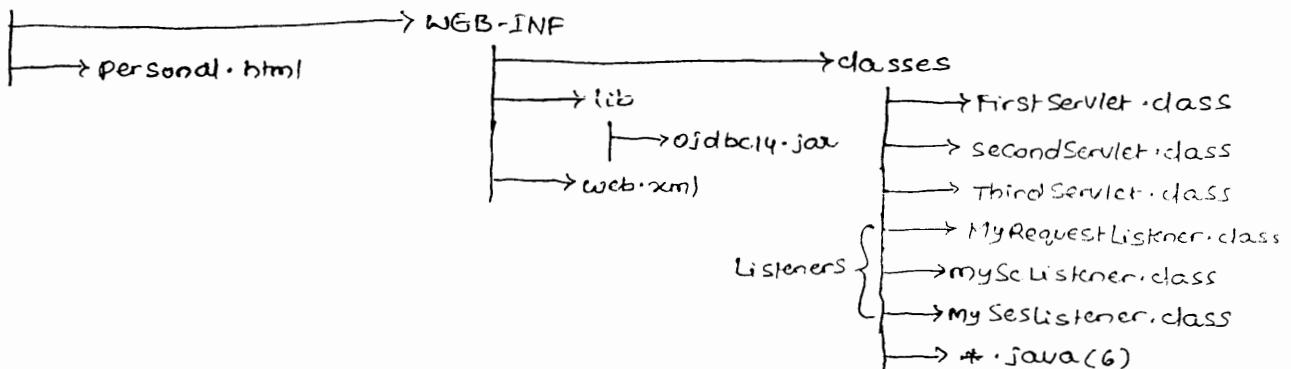
source object : HttpSession object  
Event class : javax.servlet.http.HttpSessionEvent  
Listener : javax.servlet.http.HttpSessionListener } For event handling on HttpSession object  
event handling methods : (1) sessionCreated()  
(2) sessionDestroyed()

source object : HttpSession object  
Event class : javax.servlet.http.HttpSessionBindingEvent  
Listener : javax.servlet.http.HttpSessionAttributeListener } For event handling on HttpSession attribute.  
event handling methods : attributeAdded()  
attributeRemoved()  
attributeReplaced()

\* Through Servlet Listeners we can perform event handling on request, Servlet Context, Session objects being from outside the servlet, JSP programs.

Example application (with respect to SessionApp application)

SessionApp



Every ServletListener class must be configured in web.xml file using <listener>, <listener-class> tags.

Step(1): Add Listener classes to WEB-INF classes folder of web application as shown above.

```
//MyReq.Listener.java
import javax.servlet.*;
import javax.servlet.http.*;
```

```

public class MyReqListener implements ServletRequestListener {
    Long sttime, endtime;
    public void requestInitialized(ServletRequestEvent sre) //executes when
                                                        //req obj is created.
    {
        sttime = System.currentTimeMillis();
    }
    //executes when req obj is destroyed.
    public void requestDestroyed(ServletRequestEvent sre)
    {
        endtime = System.currentTimeMillis();
        ServletContext sc = sre.getServletContext();
        sc.log("Request is processed for " + (endtime - sttime) + "ms");
        System.out.println("request is processed for " + (endtime - sttime) + "ms");
    }
}

```

### My ScListener.java

```

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

public class MyScListener implements ServletContextListener
{
    Long sttime=0;
    long endtime=0;

    public void contextInitialized(ServletContextEvent sce)
    {
        sttime=System.currentTimeMillis(); ServletContext sc = sce.getServletContext();
        sc.log("web application is deployed at " + new Date());
    }

    public void contextDestroyed(ServletContextEvent sce)
    {
        endtime = System.currentTimeMillis();
        sc.log("web application is undeployed/reloaded/stopped at " + new Date());
        sc.log("web application is there in running continuously for " + (endtime - sttime));
        System.out.println("web application is there in running continuously for " + (endtime - sttime));
    }
}

```

### MySesListener.java

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

public class MySesListener implements HttpSessionListener {
    long stime, endtime;
    public void sessionCreated(HttpSessionEvent se) {
        stime = System.currentTimeMillis();
    }
    //execute when HttpSession obj is destroyed
    public void sessionDestroyed (HttpSessionEvent se) {
        endtime = System.currentTimeMillis();
        HttpSession ses = se.getServletContext().getSession();
        //get access to servletContext obj
        ServletContext sc = ses.getServletContext();
        sc.log(ses.getAttribute("name") + " is there in the session for"
               + (endtime-stime) + "ms");
        sc.log(ses.getAttribute("name") + " is there in the session for"
               + (endtime-stime) + "ms");
    }
}
```

Step (2): compile all the listener classes.

```
>javac *.java
```

Step (3): Configure all the three listener classes in web.xml file

#### In web.xml

```
<listener>
    <listener-class> MyReqListener </listener-class>
</listener>
<listener>
    <listener-class> MySesListener </listener-class>
</listener>
<listener>
    <listener-class> myScListener </listener-class>
</listener>
```

Step(4): Test the application and also observe the log file (D:\Tomcat60\logs\localhost.2011-12-04 file)  
current date

- \* In our java web applications no servlet listener acts as default listener. That means every listener configuration is mandatory in web.xml file.
- \* Servlet listeners are useful to keep track of various operations related to request, session, ServletContext objects without disturbing the existing code of web application.
- \* To make the java class as servlet listener class the class must implement XxxListener interface.

#### Procedure to Configure Jboss 5.x server with NetBeans IDE

##### Step(1):

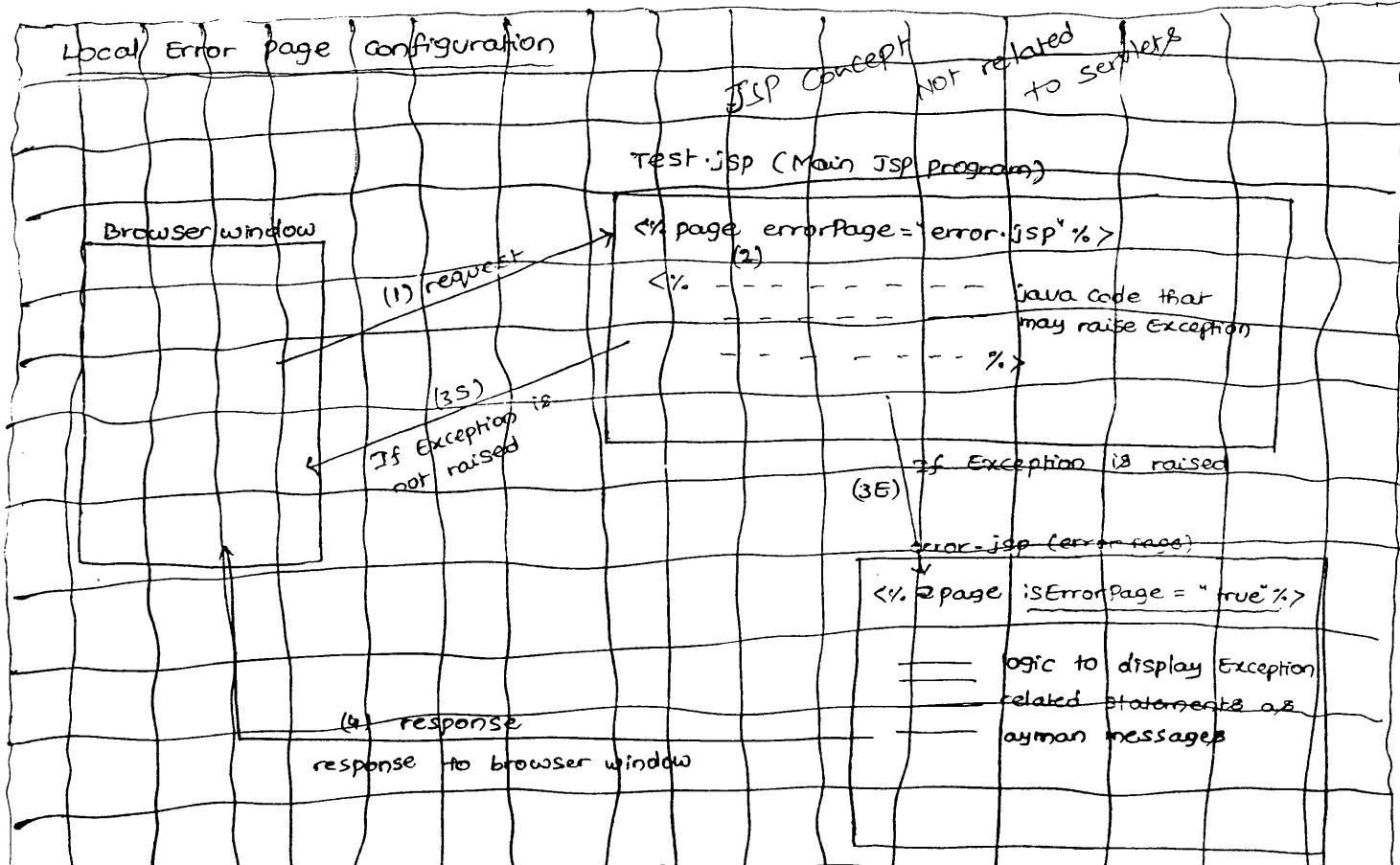
Tools → Servers → add server → Jboss application server → next →  
Server location: E:\JBoss5.x\Soft\jboss-5.1.0.GA → next →  
Domain: default → finish → close

- \* In servlet programming we can use servlet OutputStream class object pointing to response object to write the o/p of the servlet program to browser window or web page through webserver and it is alternate for PrintWriter stream object.

Ex : res.setContentType("text/html");  
ServletOutputStream sos = res.getOutputStream();  
sos.println("welcome to servlets");  
sos.println("hello");

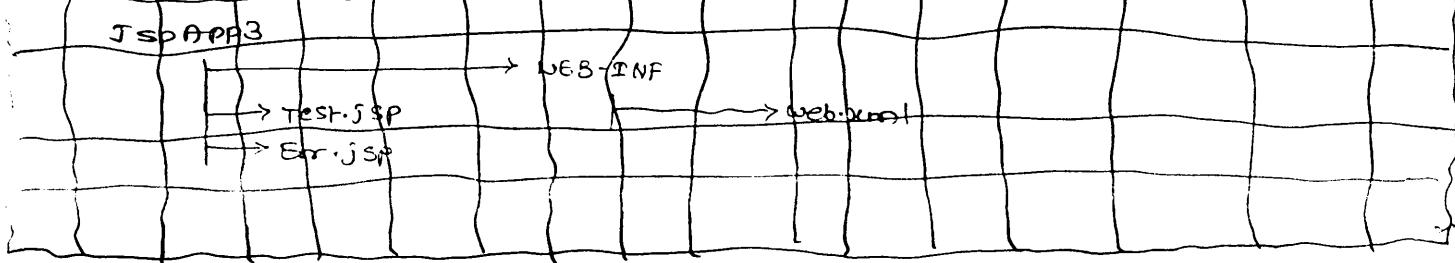
- \* In one servlet program we can not use both stream objects (Servlet OutputStream, PrintWriter) at a time pointing to res object.
- \* PrintWriter is a character stream available in java.io package and this stream is good when servlet program generates more character data and less binary data to write to browser window.
- \* ServletOutputStream is byte stream supplied by servlet api and this stream is good when servlet program generates more binary data and

- less footer data to write to browser window.
- \* for better performance of servlet and to reduce one method call use `pw.print()` method in servlet program instead of `pw.println()` method because this `pw.println()` method internally calls `pw.print()` method to complete the requirement.
- \* we can also apply this recommendation while working with `ServletOutputStream`.



- \* `isErrorPage = "true"` attribute of `page directive` tag can make a jsp program of web application as error page.
- \* The implicit object `Exception` is visible only in that JSP program that acts as error page and we can use this object to know the details of exception that are raised in main JSP programs.

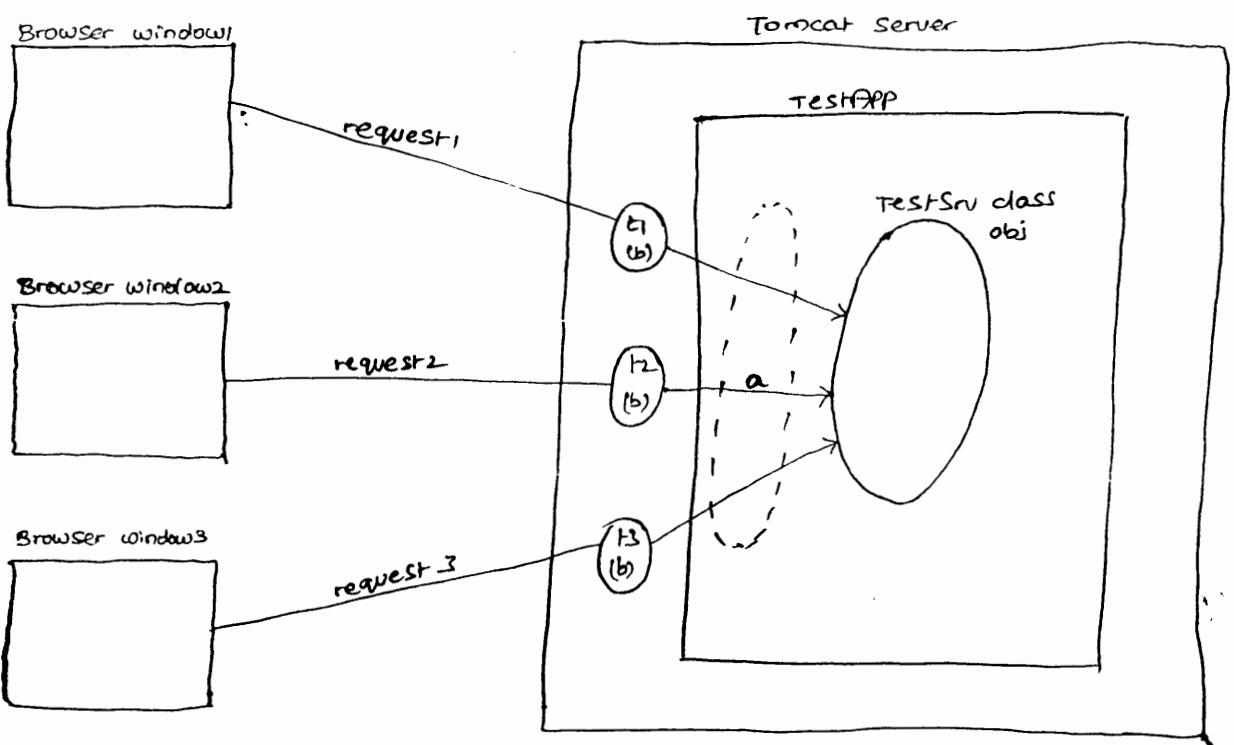
#### Example application



## Thread safety

- \* If multiple threads are running on single object or variable simultaneously or concurrently then data of the object may corrupt. This says our object is not thread safe object.
- \* To make object/ variable as thread safe apply lock on object or variable do through synchronization and allow only one thread at a time to manipulate object/ variable data. The instance variables of our server class are not thread safe by default where as the local variables declared in service(-,-) or doxxx(-,-) methods of our server class are thread safe by default.

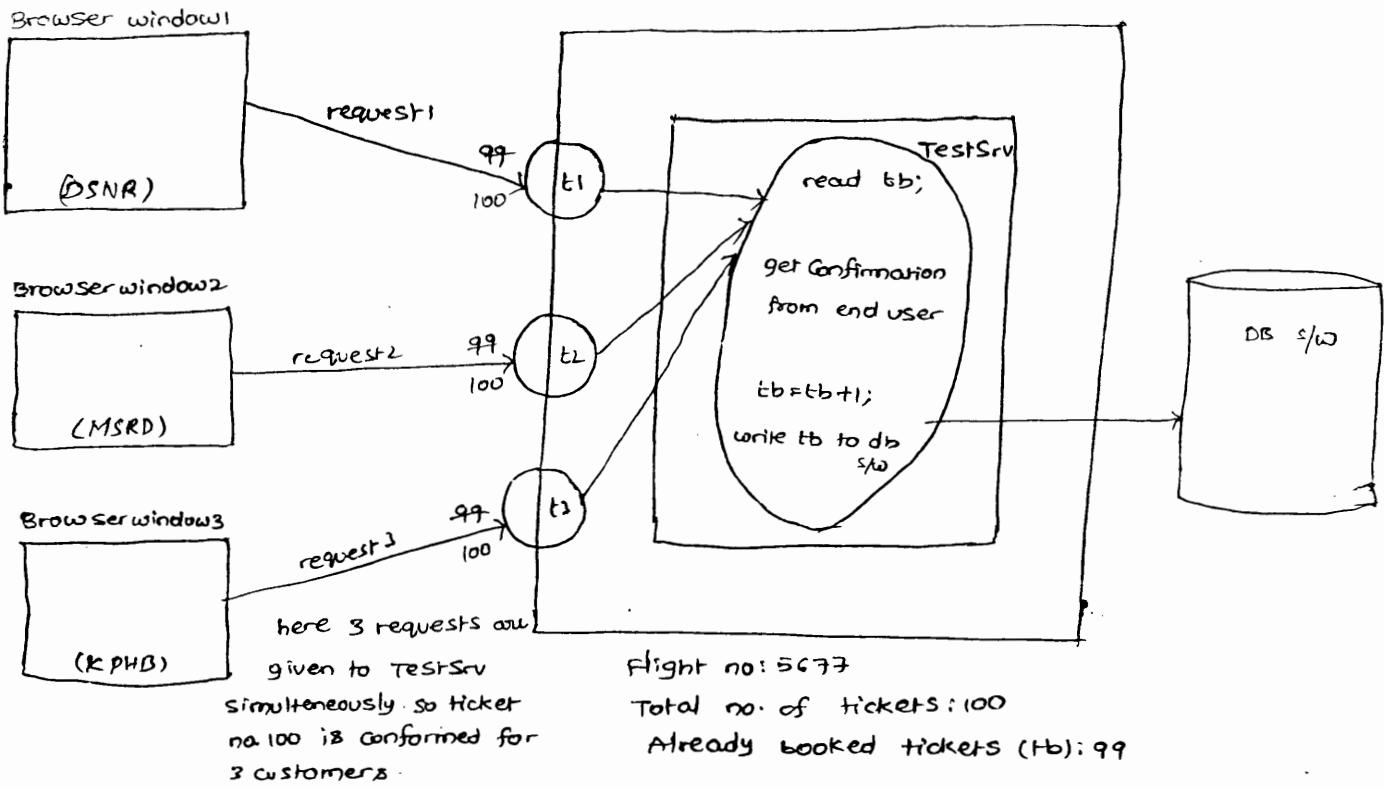
```
public class TestSrv extends HttpServlet
{
    int a;
    public void service(<-->)/doxxx(<-->) throws SE,IOE
    {
        int b;
        ...
    }
}
```



- \* In the above diagram all the three threads started on TestSrv class object will act on single copy of instance variable 'a' but every thread gets its own copy of local variable 'b'.

- \* By default every servlet program is a single instance multiple threads component so by default no servlet program is thread safe.
- \* Making servlet program as thread safe program is nothing but making the instance variables of servlet class as thread safe through synchronization and other concepts.

### understanding the problems related to working with non thread safe servlets (regular servlets)



- \* Simultaneous request or concurrent request in servlet execution is nothing but Servlet program is allowing another request from clients to execute the logic before completion of existing request processing. In such situations servlet program may generate wrong results like booking same ticket number for multiple passengers as show in the above diagram.
- \* To make our servlet program as thread safe servlet program we can use one of the following four techniques.
  - Work with only local variables of service(-) / doXXX(-) methods
  - Work with synchronized service(-) / doXXX(-) methods
  - Work with synchronized blocks in service(-) / doXXX(-) methods

(4) make your servlet class implementing javax.servlet.SingleThreadModel (I).

#### (1) working with only local variables

```
public class TestSrv extends HttpServlet
{
    == no instance variables

    public void service(--) / doxxx(--) throws SE, IOE
    {
        int a,b,c;
        Connection con;
        ===== rewrite entire using local variables
    }
}
```

\* Local variables of service(--) / doxxx(--) methods are thread safe variables by default. So the above servlet becomes thread safe servlet. Because it does not have instance variables and working with just local variables.

NOTE: This technique is not recommended to use because it is practically impossible to develop servlet classes without instance variables.

#### (2) Working with synchronized service(--) / doxxx(--)

```
public class TestSrv extends HttpServlet
{
    Connection con; } Instance variables
    int a,b,c;

    public synchronized void service(--) / doxxx(--)
    {
        ===== // write logic using instance variables
    }
}
```

\* All requests that are given to servlet starts threads on our servlet class object and executes service(--) / doxxx(--) .

\* In the above code multiple threads started on our servlet class object will act on single copy of instance variables but only one thread is allowed at a time to act on those instance variables because the service(--) method is taken as synchronized method.

\* Instead of making whole service(--) as synchronized method it is recommended to take the support of synchronized blocks on special objects on which you are looking for thread safety.

### (3) working with synchronized blocks in service(-,-) / doxxx(-,-)

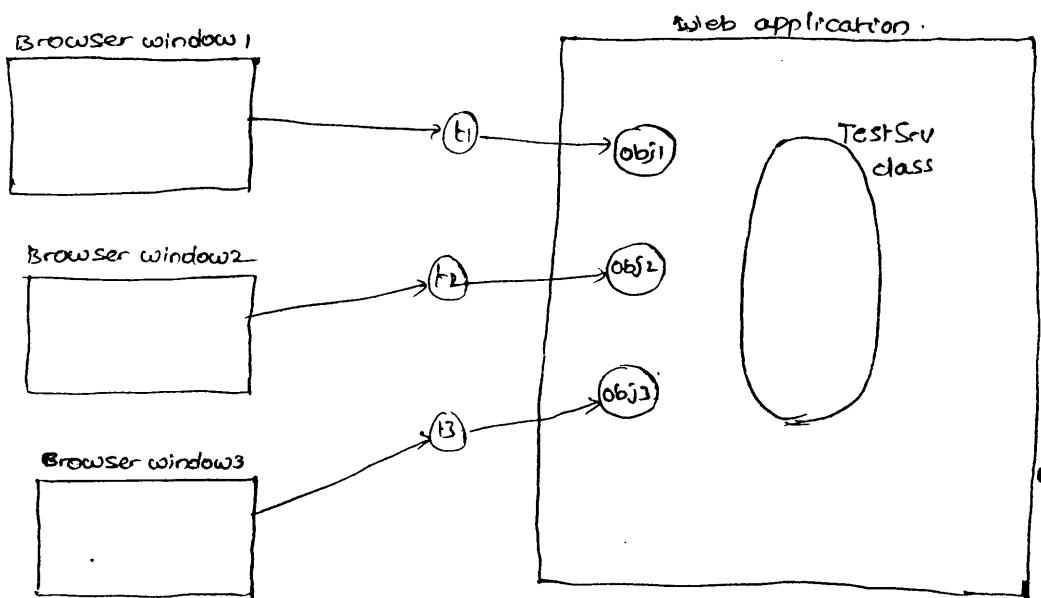
```
public class TestSrv extends Gs/HS
{
    Connection con;
    int a,b,c;
    public void service(-,-) / doxxx(-,-)
    {
        ===
        synchronized (con obj)
        {
            === logic related to con obj
        }
        synchronized (session obj)
        {
            === logic related to HttpSession obj
            === (creating, modifying, reading and writing, moving attribute values)
        }
        synchronized (ServletContext obj)
        {
            === logic related to ServletContext obj
            === (creating, modifying, reading and moving attribute values)
        }
    }
}
```

- \* working with these synchronized blocks is the most popular technique of real world to make servlet programs as thread safe.
- \* since HttpSession object and ServletContext objects are sharable objects in multiple web resource programs of web application it is recommended to use them in synchronized blocks as shown above.

### (4) Making our servlet class implementing javax.servlet.SingleThreadModel (I)

```
public class TestSrv extends HttpServlet implements SingleThreadModel
{
    int a,b;
    Connection con;
    public void service(-,-) / doxxx(-,-)
    {
        ===
        ===
    }
}
```

- \* By seeing the implementation of `singleThreadModel` interface the underlying server automatically makes our servlet as thread safe. But the technique of thread safety that is used by server will vary server to server.
- \* This interface has been deprecated from Servlet API 2.4 because different servers are using different mechanisms while implementing this interface and to make the servlet as thread safe some servers are even creating multiple objects for servlet class for multiple requests on one per request basis. This is violation of servlet specification that says a servlet is a single instance multiple threads component.



`obj1, obj2, obj3` are the objects of `TestSrv` class

- \* Every JSP program is <sup>not</sup> thread safe by default because the JSP equivalent is not servlet automatically implements `singleThreadModel` interface and <sup>not</sup> automatically generates the necessary synchronized blocks in `JspService(--)` method.
- \* To make JSP program as <sup>not</sup> thread safe program use `<%@ Page isThreadSafe="false" %>` tag in JSP program. The default value of `isThreadSafe` attribute is "true".
- \* `isThreadSafe="true"` makes JSP equivalent servlet as non thread safe servlet. where as `isThreadSafe="false"` makes JSP and its equivalent servlet as thread safe.

## File downloading

1. Making the output of web resource program as downloadable file  
(approach 1)

2. Making the resource (file) of server machine file system as downloadable file  
(approach 2)

\* In approach 1 web resource program will not be downloaded. The output of the web resource program as downloadable file through browser window. place the following the two lines of code in servlet/JSP program to make its response as downloadable file according to approach(1).

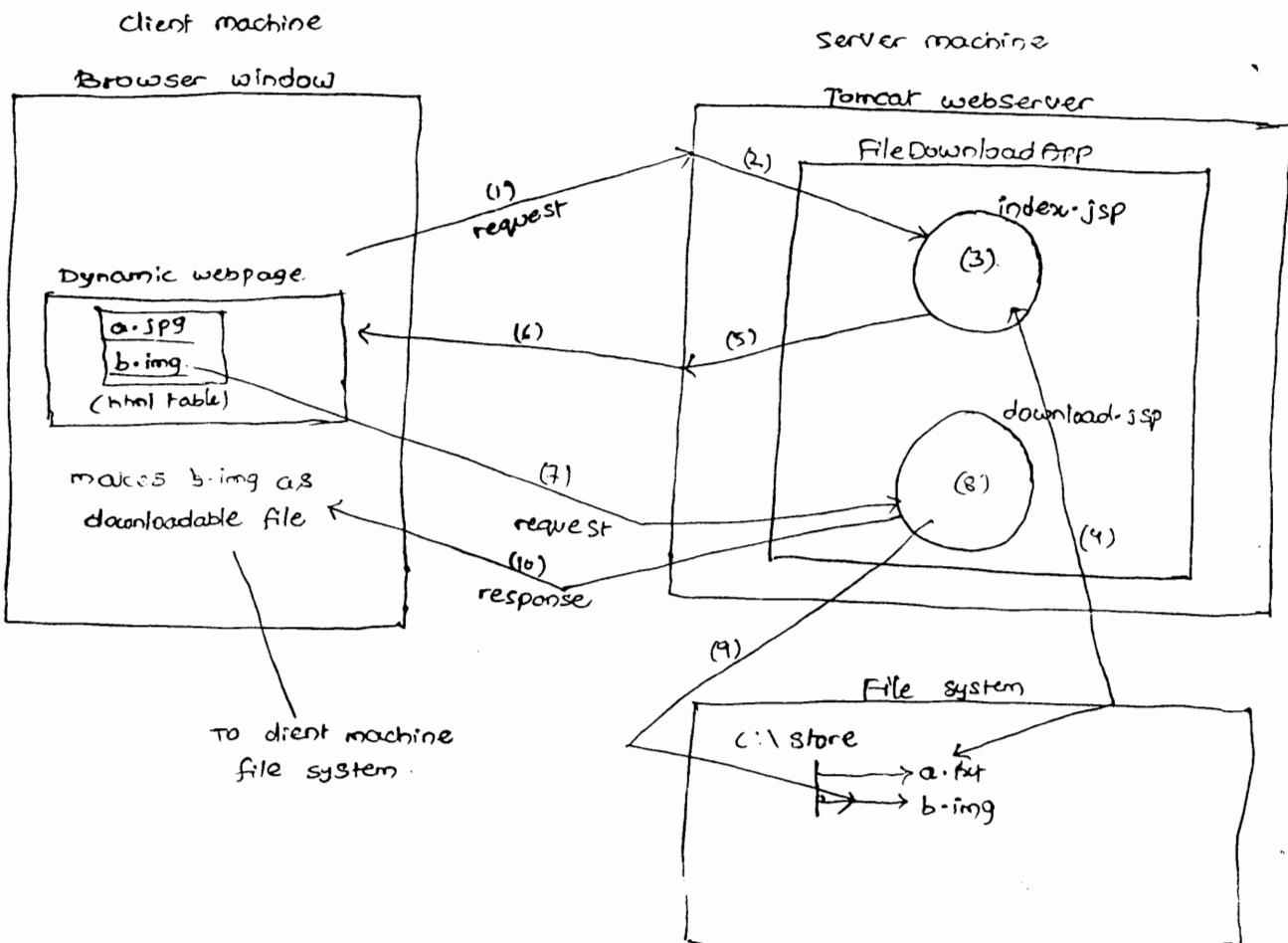
```
response.addHeader("Content-Disposition", "attachment; filename=abc.html");
response.setContentType("text/html")
```

MIME type

filename that holds response of current web resource program.

\*1 → special response header to guide webserver towards sending the response of the web resource program ~~raw~~ to client.

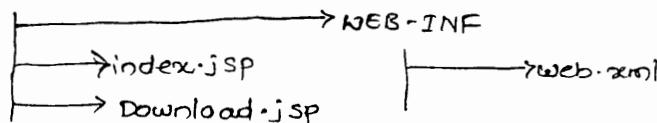
approach(2) based file downloading application



- \* The above application makes all the files of c:/store folder as downloadable files. Here index.jsp responsibility is to render all the files of c:/store folder as hyperlinks to client and download.jsp responsibility is to download specific file of c:/store folder to client machine file system.
- \* java.io.File class object can represent a file or directory of file system.
- \* The listFiles() method of java.io.File class gives all the files and sub directories of certain folder in the form of java.io.File class object array [file[]]

### Deployment directory structure

FileDownloadApp



Request URL to test the application

<http://localhost:2020/FileDownloadApp/index.jsp>

### Source code

#### index.jsp

```

<%@ page import="java.io.File,java.util.*" %>
<H1> List of All files under c:\store </H1>
<%
//locate folder on server machine file system
File dir=new File ("c:\\store");
File[] lf=dir.listFiles(); //give all files and subdirectories of c:\\store
//add all the file names of c:\\store to al obj (ArrayList obj)
ArrayList al=new ArrayList();
for(int i=0; i<lf.length; i++)
{
  if(lf[i].isFile())
  {
    al.add(lf[i].isFile(), lf[i].getName());
  }
}
%>
  
```

```

//logic to display the file names of c://store folder as downloadable
//files(hyperlinks based)

if(al.size()!=0)
{
%>
<table border=1><tr><th><b>Filename</b></th></tr>

<%
String fname=null;
for(int i=0; i<al.size(); i++)
{
    fname=(String)al.get(i);
%>
<tr>
<td><a href="Download.jsp?filename=<%=fname%>">
<%=fname%> </a></td></tr>
%>} //for
%>
%>
</table>

```

### web.xml

```
<web-app>
```

### download.jsp

```

<%@page import="java.io.*"%>

<%
//reading req param value (filename)
String filename=request.getParameter("filename");
//locate the file to be downloaded
File f=new File ("C://store//"+filename);
//get servletOutputStream object
ServletOutputStream op=response.getOutputStream();
//get mime type of the file to be downloaded dynamically
String mimetype=application.getMimeType("C://store//"+filename);
//set the mime type as response content type
response.setContentType((mimetype!=null)?mimetype:"application/octet-
stream");

```

```

    //set response content length
    response.setContentType("text/html");
    //work with special header to make file as downloadable file.
    response.setHeader("Content-Disposition", "attachment; filename=" + 
                        fileName);
    //Stream and buffer based logic to download the file
    byte[] buf = new byte[1024];
    int length=0;
    DataInputStream dis=new DataInputStream(new FileInputStream(f));
    while((dis!=null)&&(length=dis.read(buf))!=-1)
    {
        op.write(buf,0,length);
    } //while
    dis.close();
    op.flush();
    op.close();
%>

```

### Security in servlets :

\* security = Authentication + Authorization

\* security means authentication + authorization.

\* checking the identity of a user is called as authentication.

\* checking the access permissions of a user to use certain resources of the application is called as authorization.

Ex: Every employee must be authenticated to use banking project. But only cashier is authorized to use cash module. Administrator is authorized to use every module.

\* Programmers are not responsible to perform network security. These operations will be taken care by network administrator through firewalls.

\* Programmers are responsible for application managed security and server/container managed security in web applications.

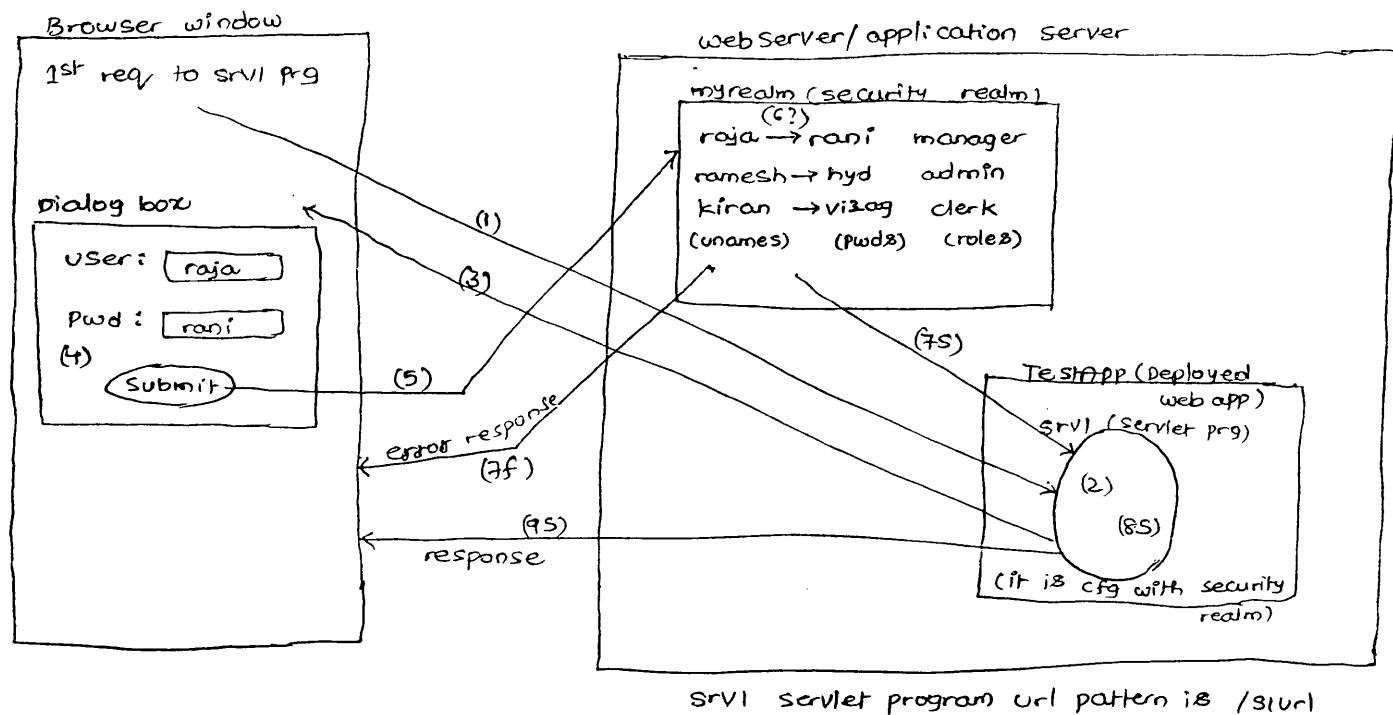
\* Application managed security means it is the security that is developed by programmer manually inside the application.

\* Server managed security means it is the authentication and authorization performed by container on web resource programs by using security middleware.

service.

\* In server managed security the logical context where users, passwords and roles are defined is called as security realm and every server comes with one default security realm called myrealm.

\* In tomcat server this default realm related configurations can be done in <tomcat-home>\conf\tomcat-users.xml file.



\* A role defines access permissions (set of access permissions) instead of specifying access permission for each user separately define a role specifying access permissions and assign that role to user.

with respect to diagram

(1) End user gives direct request to svrl program.

(2)(3) Since no user name and password is coming along with the request and since svrl is configured with security realm the svrl sends 401 response status code based response having one dialog box.

(4)(5) End user submits dialog box having username and password and that request will be traped by security realm.

(6)(7) Security realm verifies whether given username, password correct or not against the username and password of realm.

(8) If correct, request goes to svrl program.

(9) If not correct it renders same dialog box for two more times then shows 403 status code based error response.

(8s)(s) server program executes and generated response goes to browser window.

\* We can configure server managed security authentication on our web resource program in four modes.

(1) BASIC

(2) DIGEST

(3) FORM

(4) CLIENT-CERT

BASIC → uses base 64 algorithm internally.

\* does not encrypt given username and password.

\* Generates built-in dialog box for end user to submit username and password.

DIGEST :

\* uses MD5 hashing algorithm internally.

\* It is stronger than base-64 algorithm.

\* Remaining all are same as BASIC, but it encrypts given username and password.

FORM :

\* same as BASIC but allows programmer to design his own form page, error page instead of ready made dialog box, ready made errors message.

CLIENT-CERT :

\* works with digital certificates using some algorithms like RSA.

\* use https and SSL concepts to implement this.

\* To configure security realm and authentication models in the web resource programs of java web application use various configurations in web.xml file.

\* For example code on, BASIC, DIGEST - FORM based authentication refer supplementary handout given on 25-12-2011.

Procedure to create users and roles in default security realm (myrealm)  
of Tomcat server

Go to <Tomcat-home>\conf\tomecat-users.xml → add these entries.

& write rolesName

```
<user username = "raja1" password = "raja1" roles = "manager"/>
```

```
<user username = "raja2" password = "raja2" roles = "admin"/>
```

```
<user username = "raja3" password = "raja3" roles = "manager,admin,tomcat"/>
```

- \* Uncomment existing roles and users
- \* This server managed security can be used as outer layer security for web application even though web application contains application level security. This is very useful in military based, NASA based, ISRO based web applications.
- \* The client-cert authentication model sends digital certificates to browser window as the response of initial request and that digital certificate will be installed at client side with user's permission. Now onwards browser window and server communicates with each other having encrypted data based that is generated by algorithm that is used to generate digital certificate. In this browser window (client) and server should communicate with each other by using HTTPS protocol (HTTP over SSL).
- \* In CLIENT-CERT authentication model server allows only those clients which send request having digital certificate.
- \* To make server sending digital certificate to browser as the response of initial request, the programmers must develop and configure digital certificate with server
- \* In java one built-in tool 'keytool' is given to generate digital certificates using different algorithms.

Procedure to create digital certificate, to connect digital certificate with server and to work with CLIENT-CERT authentication model using HTTPS Protocol

Step(1): Generate digital certificate using "keytool" by specifying RSA algorithm

```
> keytool -genkey -alias tomcat1 -keyalg RSA  
                                ↓  
                                any name (logical name of digital certificate)
```

Enter key store password: sathyal

Re-enter new password : sathyal

: correct?

yes

Enter key password for <tomcat1> ; (press Enter key)

(2): The above steps based digital certificate will be generated in c:\Documents and setting \ welcome folder as .key store file.  
↓  
currently logged in windows username

Step(3): Configure the above generated digital certificate with Tomcat server by enabling HTTPS protocol.

Go to <Tomcat-home>\conf\server.xml file → un comment that <Connector> tag that points to SSL HTTP/1.1 protocol (line no. 66) → make sure that the <Connector> tag at line 66 (3rd <Connector> tag) is having following content

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"  
maxThreads="150" scheme="https" secure="true" clientAuth="false"  
sslProtocol="TLS" keystoreFile="c:\Documents and Settings\welcome\keyStore"  
keyStorePass="satyal"/>
```

↓  
The above generated digital certificate file

↓  
The above chosen password.

Step(4): Restart the server.

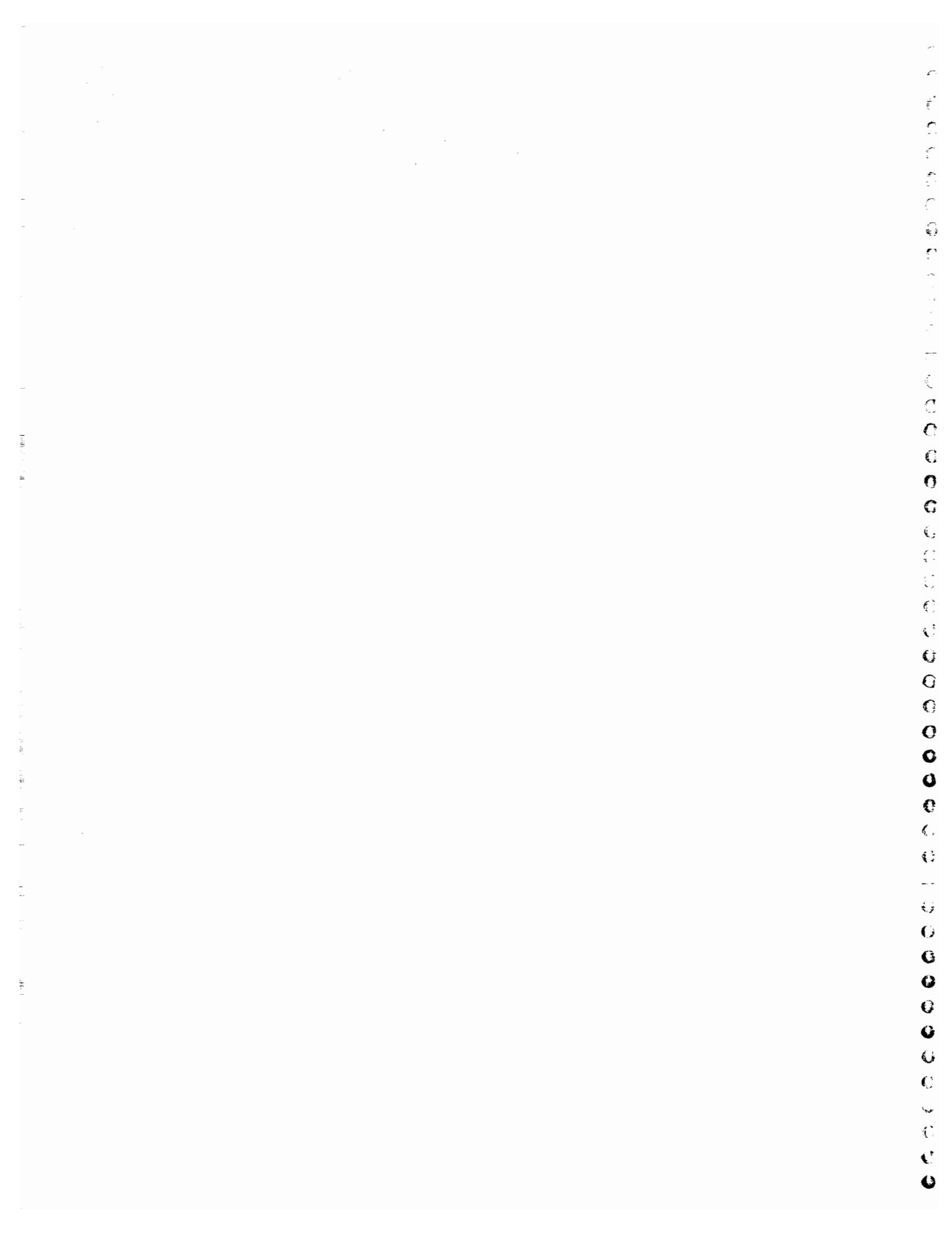
Step(5): Give request to tomcat server from browser window.

https://localhost:8443  
https://localhost:8443/wfOne/input.html  
Protocol ↓  
proto port no. of http with ssl environment

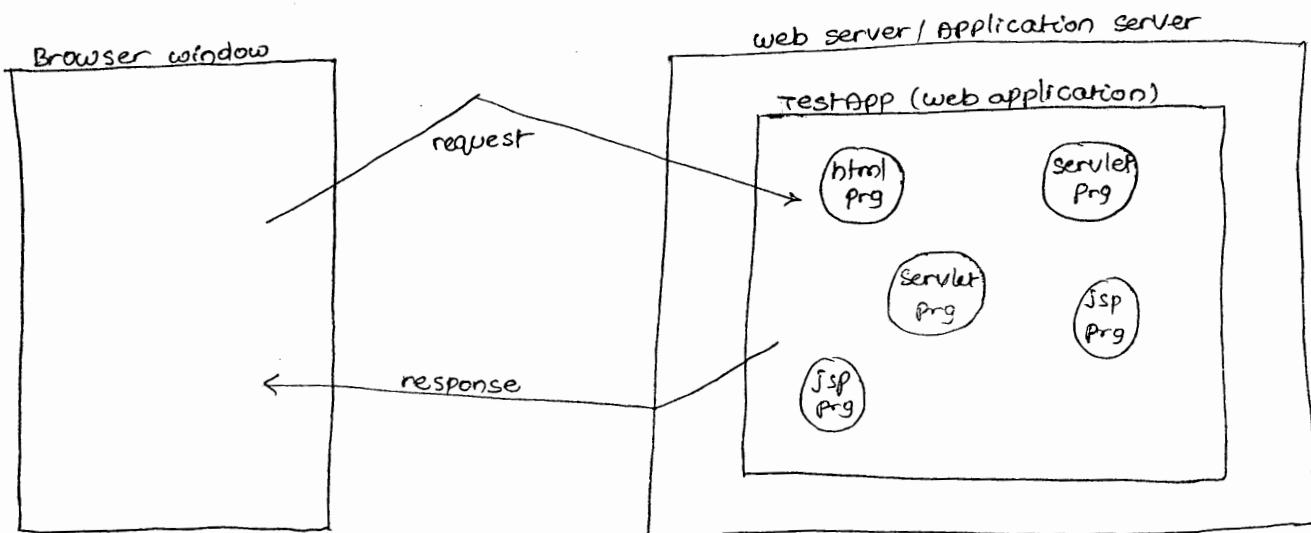
\*For related information on SSL refer the SSL chapter of tomcat documentation.

NOTE: Industry prefers working with either form based or CLIENT-CERT models for securing web applications.

When CLIENT-CERT is enabled we can even also enable Basic or DIGEST or FORM authentication models.



# JSP



\* A web application is a collection of web resource programs having capability to generate web pages. There are two types of web resource programs  
(1) Server side programs  
(2) Client side programs

\* Server side programs resides and executes from the server.  
Ex: servlet program, JSP program and etc.

\* Client side programs resides in web server but they come to browser window for execution.

Ex: html program, Javascript program, VB script program and etc..

NOTE: Decide whether web resource program is client side or server side program based on the place where it executes not based on the place where it resides.

\* Server side programs generate dynamic web pages.

\* Client side programs generate static web pages.

\* Every web application is a collection of both client side and server side web resource programs.

Client side technologies (useful to develop clientside web resource programs)

html

javascript

VB Script

Ajax

## Server side technologies (useful to develop serverside web resource programs)

Servlets → from sun ms (2) } (1)  
JSP → from sun ms (3)  
ASP → from microsoft  
asp.net → from microsoft (4)  
PHP → from Adobe & Apache foundation (5)  
SSJS → from NestCape  
ColdFusion → from Adobe

why sun microsystem has given JSP as server side technology when they have already got servlets as server side technology?

(A) while working with ASP, PHP technologies we can go for tags based programming. To work with servlets strong Java knowledge is required and tags based programming is not possible so no programmer had liked servlets in its initial days. To attract non-Java programmers like ASP programmers the Sun microsystem has given a tag based server side technology called JSP having all the features of servlets.

### Drawbacks of servlets Programming

(1) Strong Java knowledge is required so not suitable for non-Java programmers.

(2) The presentation logic (HTML code) will be mixed up with Java code (pw. `println()`) so modification done in one code may affect another code.

(3) Writing HTML code in servlet programming is a complex process.

(4) In servlet programming implicit objects are there but we need to write some additional code to get access to those objects.

(5) Modification done in source code of servlet program will be effected only after compilation of servlet program and reloading of web application in most of the servers.

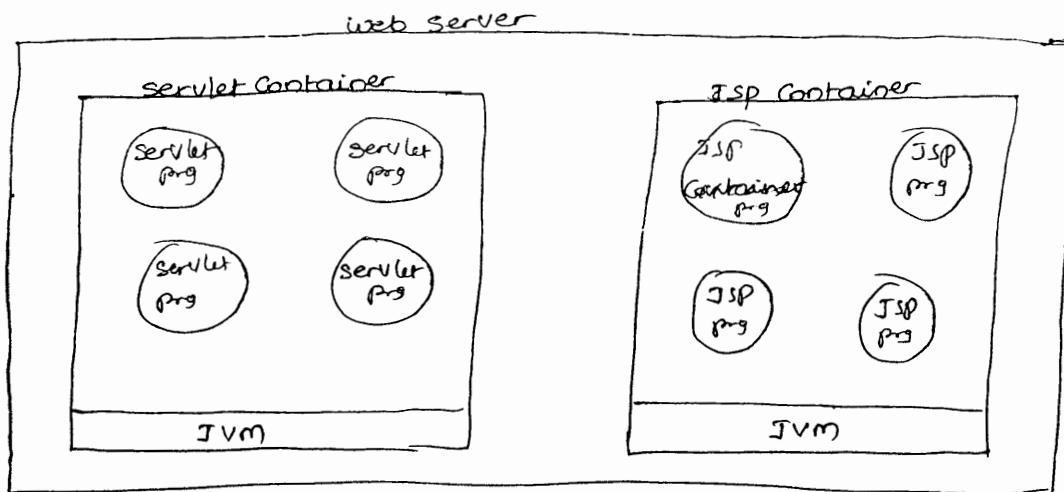
(6) Programmer should explicitly take care of exception handling.

→ The session object required for session tracking should be created by the programmer manually.

(7) Learning curves quite high in servlets & servlet programming is not thread safe by default so we need to write some additional code to make our servlet program as thread safe.

### Features of JSP

- (1) Allows tag based programming so strong java knowledge is not required.
- (2) suitable for both java and non-java programmers.
- (3) use 9 implicit objects and we can use them directly in our JSP program.
- (4) Modifications done in JSP program will be recognized by underlying server automatically without reloading of web application.
- (5) takes care of exception handling.
- (6) JSP program is thread safe by default.
- (7) allows us to separate presentation logic from (HTML code) java code (business logic)
- (8) Easy to learn and utilize.
- (9) Allows us to work with JSP supplied built-in tags and third party supplied JSP tags and even allows to develop custom JSP tags.
- (10) use all the features of servlet.



- \* JSP, ASP, ASP.NET, PHP programs are there to execute based on page compilation process.
- \* The process of converting one form of program source code to another form of source code is called as page compilation.
- \* In JSP page compilation the source code of JSP program will be converted into an equivalent servlet program source code. For this JSP page compiler is required.
- \* Every web server/application server supplies one JSP page compiler.
- \* Servlet Container executes Servlet programs and takes care of the life cycle of Servlet programs.

- \* JSP Container executes JSP programs by executing the JSP equivalent servlet programs and also takes care of the life cycle of JSP programs.
- \* Every JSP container is enhancement of Servlet Container.
- \* Every web server and application server supplies Servlet Container and JSP Container.
- \* The Servlet Container is developed based on Servlet API specification. In Tomcat Server Servlet Container name is catalina.jar.
- \* JSP Container will be developed based on Servlet and JSP API specification. In Tomcat Server JSP Container name is Jasper (Tomcat-home\lib\jasper.jar)

### The 9 implicit objects of JSP programming

out

config (It is ServletConfig)

application (It is ServletContext obj)

request

response

page

pageContext

exception

session

- \* Once the JSP equivalent servlet is generated all these objects will be created in that servlet automatically. So we can call them as implicit objects or built-in objects of JSP.

### What is the difference between HTML and JSP?

#### HTML

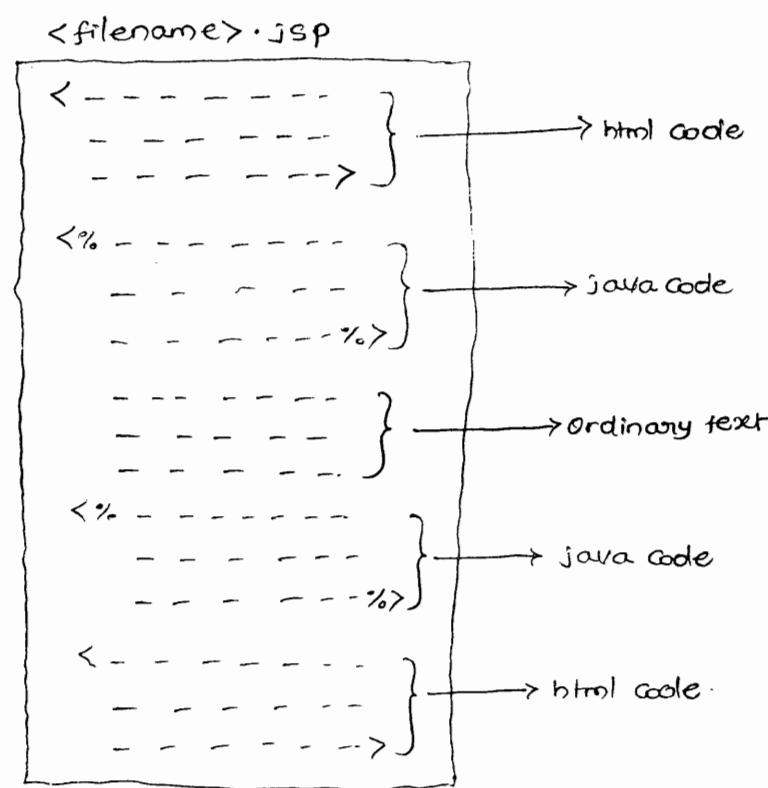
- \* Client side technology
- \* Needs HTML parser interpreter for execution.
- \* HTML program generates static web pages.
- \* Given by W3C.
- \* Can be used in any kind of web application development (Java & non-Java web applications).

#### JSP

- \* Server side technology.
- \* Needs JSP page Compiler and JSP Container for execution.
- \* JSP programs can generate static & dynamic web pages.
- \* Given by Sun Microsystems.
- \* Should be used only in Java based web applications.

\* Does not allow to create custom tags. \* Allows us to create custom JSP tags.

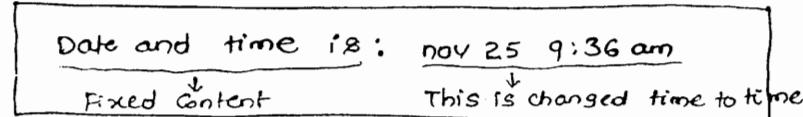
### Structure of JSP program



\* <% - - - %> is called as scriptlet.

Template text = html code + ordinary code (text)

### Dynamic web page (Generated by JSP program)



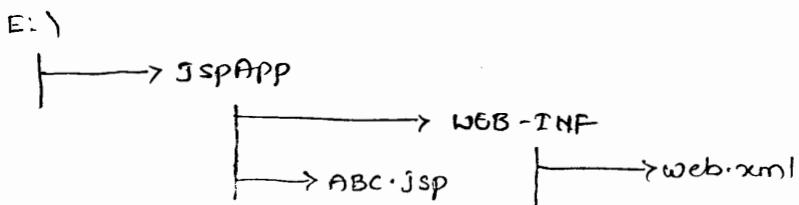
\* JSP program can generate dynamic web page. The fixed content of this dynamic web page will be generated through template text of JSP program. Similarly the dynamic values/ content of that dynamic web page will be generated through java code of scriptlet placed in JSP program.

### Ex code ABC.jsp

```
<b> <i> Date and Time is: </i> </b> → Template text
<% java.util.Date d=new java.util.Date();
   out.println(d.toString()); %> → Java code
   ↓
   implicit object.
```

## Procedure to deploy the Jsp program based web application

Step(1): create deployment directory structure by taking the JSP program as web resource program.



### ABC.jsp

same as above code.

### web.xml

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

NOTE: Configuration of JSP program in web.xml file is optional.

Step(2): Deploy the above web application in tomcat server.

copy E:\JspApp folder to <tomcat\_home>\webapps folder.

Step(3): Start the tomcat server.

Step(4): Test the webapplication. Give request to ABC.jsp

open browser window → type this url

```
http://localhost:2020/JspApp/ABC.jsp
```

\* while working with only JSP programs based web applications the web.xml file is optional in most of the servers but it is recommended to place web.xml file.

\* we can hide the JSP program file name & extension from end users to hide technologies details from end users by providing url pattern to JSP program in web.xml file.

\* we can provide url-pattern to JSP program by containing Jsp program in web.xml file as shown below.

```
<web-app>
```

```
  <servlet>
```

```
< servlet-name> xyz </servlet-name>
    <jsp-file> /ABC.jsp </jsp-file>
</servlet>
<servlet-mapping>
    <servlet-name>xyz </servlet-name>
        <url-pattern> /test1 </url-pattern>
    </servlet-mapping>
</web-app>
```

\* Using following request urls to generate request to JSP program.

http://localhost:2020/JspAPP/ ABC.jsp

http://localhost:2020/JspAPP/ test1

\* WEB-INF is a private directory of java web application so only web server can recognize that directory.

\* The web resource programs that are placed inside inside WEB-INF (or) its sub folders will be recognized by the web server only when they are configured in "web.xml" file.

Ex: servlet programs.

\* The web resource programs that resides outside the WEB-INF folder can be recognised by the webserver directly so there is no need of configuring those programs in web.xml file.

Ex: html programs, JSP programs.

\* The JSP program can have two types of objects.

(1) Implicit objects:

The 9 implicit objects like out, request, response and etc..

(2) Explicit objects

Objects created by programmer manually.

Ex:

```
<%. java.util.Date d = new java.util.Date(); %>
    ↓
Explicit object
```

\* this, super key words are implicit objects of stand alone java programs.

\* request, response, ServletConfig & ServletContext objects are implicit objects of servlet programming and we need to write code to get access to those objects.

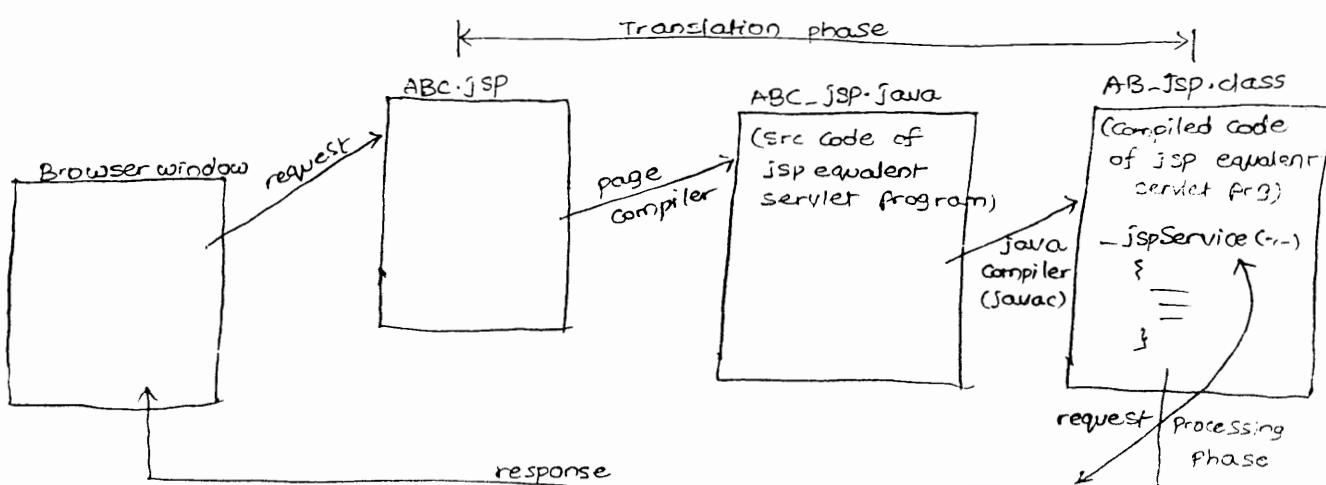
\* In the execution process of JSP program two phases are there.

1) Translation phase

2) Request Processing phase

\* In translation page the JSP program will be converted into an equivalent servlet program.

\* In Request Processing phase the compiled code of JSP ~~as~~ executes and the generated response goes to browser window.



\* In Request Processing phase the `-jspService(..)` of JSP equivalent servlet executes and generated response goes to browser window as web page.

### The life cycle methods of Servlet program

`init(ServletConfig config)`

`service(ServletRequest req, ServletResponse res)`

... etc ...

## The life cycle method of JSP program

`jspInit() / - jspInit()` ①, ② are given from tomcat 6 onwards and not there in other servers as life cycle methods.

`-jspService(HttpServletRequest req, HttpServletResponse res)`

`jspDestroy() / - jspDestroy()` ③

\* `-jspInit()`, `-JSPDestroy()` methods are introduced as life cycle methods from recent versions of JSP so only latest web server s/w and application server s/w's are supporting them.

\* `jspInit() / - jspInit()` life cycle methods execute when JSP equivalent Servlet class is instantiated.

\* `-jspService(-,-)` life cycle method executes when JSP program gets request from clients.

\* `jspDestroy()` life cycle method executes when the servletContainer is about to destroy object of JSP equivalent servlet program (servlet class).

\* In tomcat server the JSP equivalent servlet program of ABC.jsp belonging to JSPApp web application comes in `<tomcat-home>\work\catalina\localhost\JSPApp\org\Apache\JSP` folder as ABC-JSP.java, ABC-JSP.class.

package name of JSP  
equivalent servlet class

↓  
Source code of JSP equivalent servlet

→  
Compiled code of JSP  
equivalent servlet

\* The file name and class name of JSP equivalent servlet program changes server to server.

\* The generated JSP equivalent servlet is HttpServlet.

\* JSP Container internally takes the support of servlet container and servlet life cycle methods call JSP life cycle method and to execute JSP equivalent servlet program.

\* Every JSP equivalent servlet extends from one JSP container supplied Java class and this Java class extends from HttpServlet class and overrides servlet life cycle methods calling JSP life cycle methods internally.

\* In Tomcat server JSP equivalent servlet class extends from the Jasper (JSP container) supplied org.apache.jasper.runtime.HttpJspBase class and this HttpJspBase class extends from HttpServlet class, overrides servlet life cycle methods calling JSP life cycle methods internally.

\* `init(-)` → internally calls `-jspInit() / jspInit()`

\* `service(-,-)` → internally calls `-jspService(-,-)`

\* `destroy()` → internally calls `-jspDestroy() / jspDestroy()`

- \* refer source code of JSP equivalent servlet and HttpServletBase class once.
- \* In Tomcat server jasper.jar represents JSP container.
- \* In weblogic server weblogic.jar represents JSP container, EJB container, servlet container and etc...
- \* To deploy above given JSPAPP web application in Adv-javaBatch Domain of weblogic 10.3

copy JSPAPP folder to <weblogic\_home>\middleware\user\_projects\domains\Adv-javaBatchDomain\autodeploy folder.

- \* To generate request to JSPABC.jsp of above JSPAPP webapplication use the following request URL.

<http://localhost:7001/JSPAPP/ABC.jsp>

- \* JSP page Compiler name in weblogic is JSPC and it generates ABC.jsp related JSP equivalent servlet as shown below.

--abc.java (source code JSP equivalent servlet program)                          } For location  
 --abc.class (compiled code of JSP equivalent servlet program) } Search in subfolders  
 of Adv-javaBatchDomain

NOTE: This JSPC pageCompiler deletes source code of JSP equivalent Servlet program once compiled code is generated.

- \* The request given to JSP program participates only in request processing phase when JSP program source code is not modified compare to previous request and .class file of JSP equivalent servlet program is not deleted otherwise the request given to JSP program participates in both translation phase, request processing phase.

- \* JSP Container preserves the source code of JSP program internally related to current request until next request comes to that JSP program for compilation purpose. In this process JSP container internally uses ARAXIS JDIFF kind of compilation tools internally.

There is no necessity of Configuring JSP equivalent servlet program generated by JSP page Compiler in web.xml file.

- All implicit objects of JSP will be created automatically in \_JspService(..) method of JSP equivalent servlet program.

Tomcat server gives servlet-api.jar, JSP-api.jar files representing the Sun micro system supplied Servlet, JSP api specifications. The same Tomcat server gives catalina.jar (as servlet container), jasper.jar (as JSP container)

Based or that are developed based on servlet, JSP api specifications.  
what happens if we modify the source code of JSP equivalent servlet  
program manually ?

- (A) The modifications will be effected in the output of JSP program when  
(i) JSP equivalent servlet program is recompiled.  
(ii) when the web application is reloaded.  
If request is given to JSP program without modifying source code.  
otherwise the modifications done in source code of JSP equivalent servlet  
program will not be effected.

NOTE: Compile the JSP equivalent servlet program generated by page compiler  
of Tomcat server add the following four jar files to classpath.

jsp-api.jar, servlet-api.jar, jasper.jar, el-api.jar

Available in `<tomcat-home>\lib` folder.

what is the use of enabling <load-on-startup> on JSP program (or) How  
to make first request given to JSP program directly participating in  
request processing phase?

- (A) Enable `<load-on-startup>` on JSP program Configuration done in `web.xml` file  
which makes JSP Container and Servlet Container complete translation phase &  
and to perform pre instantiation and pre initialization on JSP equivalent  
servlet program either during server startup or during deployment of web  
application
- \* If you modify source code of JSP program before first request then  
the entire effect of `<load-on-startup>` will be wasted.

To enable <load-on-startup> on JSP Program

```
<web-app>
  <servlet>
    <servlet-name>x</servlet-name>
    <jsp-file>/ABC.jsp</jsp-file>
    <load-on-startup>4</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>x</servlet-name>
    <url-pattern>/test1</url-pattern>
  </servlet-mapping>
</web-app>
```

Can you generate JSP equivalent Servlet program outside the web server (or application server)

before actually deploying the web application and requesting the web application?

(A) Yes we can do by running the `jsppageCompiler` tool outside the server at command prompt. This process is useful to know the syntactical errors of JSP program out side the server.

The WebLogic server supplies "weblogic.jspc" tool as JSP page compiler tool.

## Example on weblogic.jspc tool

Step(1): Add <weblogic-home>\middleware\wlserver-10.3\server\lib\weblogic.jar file to classpath.

Step(2): Arrange -jsp program

E:\temp  
|→ ABC.jsp

Step(3): Use weblogic.jspc tool

```
e:\temp>java weblogic.jspc -compiler javac ABC.jsp
```

If deletes `-abc.java` after generating `-abc.class` (is equivalent Servlet Prog).

The compiler that has to be used to compile the source file JSP equivalent servlet program.

```
e:\temp>java weblogic.jspc -keepgenerated -Compilerjavac AB.jspc
```

It preserves `--abc.java` even after generating `--abc.class` file.

\* Tomcat server gives org.apache.jasper.JspC class as jsp page Compiler.

Example to work with org.apache.jasper.JspC tool

(1) Keep your web application ready as shown below:

```

graph LR
    E["E:\TEMP"] --> JSPAPP["JSPAPP"]
    JSPAPP --> WEBINF["WEB-INF"]
    JSPAPP --> ABC["ABC.jsp"]
    WEBINF --> WEBXML["web.xml"]
  
```

Make sure that the following jar files are added to classpath.

③ Run the tool.

!!(Temp)\Tomcat\java org.apache.jasper.JSPC -d . ABC.jsp

This tool is capable of just generating destination folder to generate JSP source code of JSP equivalent servlet program equivalent Servlet Program.

## JSP tags

- \* The element name that is surrounded with "<", ">" symbol is called tag.
- <B> ---> Bold tag name
- B -----> Bold element name.
- \* Java code placed in JSP is called as script code. javascript code, VB script code, place in HTML program is called as script code.
- \* The code that can't be executed independently and should be embeded with other technologies based programs for execution is called as script code.

## (Tags) Built-in tags/elements of JSP programming

### Scripting elements/tags

- ① scriptlet (<%---%>)
- ② declaration (<%!---%>)
- ③ Expression (<%=---%>)

\* Scripting tags are there to allowing the programmer to place java code.

### Directive tags/elements

- ① Page Directive (<%@ page attributes %>)
- ② Include Directive (<%@ include attributes %>)
- ③ taglib Directive (<%@ taglib attributes %>)

\* Directive tags are given to provide global information to JSP page.

### JSP comments

<%-----%> (for Commenting JSP tags)

### Standard Action tags

```
<jsp:forward>
<jsp:include>
<jsp:useBean>
<jsp:setProperty>
<jsp:getProperty>
<jsp:param>
<jsp:plugin>
<jsp:fallback>
:
:
:
```

\* Standard Action tags are given to perform dynamic operations in JSP Programming at request processing phase.

### scriptlets

\* Every JSP tag can be used with two types of syntaxes.

(1) Standard syntax

(2) XML syntax

### Standard syntax of scriptlet

```
<%  
    -- -- -- } java code  
    -- -- -- %>
```

### XML syntax of scriptlet

```
<jsp:scriptlet>  
    -- -- -- } java code  
</jsp:scriptlet>
```

\* The java code placed in scriptlet goes to `_jspService(--)` method of JSP equivalent servlet program as it is.

\* Programmers can place business logic or requesting processing logic code in the scriptlets of JSP program.

\* Implicit objects of JSP are visible in the scriptlet. In one JSP program we can keep multiple scriptlets.

\* Variable declared in JSP comes as local variables of `_jspService(--)` method in JSP equivalent servlet so they must be initialized.

Ex: In ABC.jsp (with standard syntax)

```
<% int a=10;  
    int b=20;  
    int c=a+b;  
    out.println ("Result is :" +c);  
%>
```

In ABC.jsp (with the support of XML standard)

```
<jsp:scriptlet>  
    java.util.Date d = new java.util.Date();  
    out.println (d.toString());  
</jsp:scriptlet>
```

Ex(1): <jsp:scriptlet>

```
int a=10;
int b=20;
if (a<b)
    out.println("a is small");
else
    out.println("b is small");
</jsp:scriptlet>
```

\* The above code execution gives exception because in (a<b) statement "<" symbol will be taken as beginning of subtag having XML or JSP meaning. That means it will not be taken as the conditional operator of java code.

solution(1): (work with standard syntax)

```
<% int a=10;
   int b=20;
   if (a<b)
       out.println("a is small");
   else
       out.println("b is small");
%>
```

solution(2): (work with XML syntax along with <![CDATA[ . . ]]>)

```
<jsp:scriptlet>
<![CDATA[ int a=10;
           int b=20;
           if (a<b)
               out.println("a is small");
           else
               out.println("b is small");
%>
```

(\*) This "CDATA" makes page compiler to take the body of the tag as text information (java code) and suppresses the meaning of XML and JSP for that code.

NOTE: while working with XML syntax based JSP tags use "<" symbol carefully but there are no problems with ">" symbol.

```
<% public int sum (int x, int y)
      {
          return x+y;
      } %>
```

The above java code is invalid as Java does not support nested method definition. But we can place method calls code in scriptlets.

```
<% String s="Hello";
   int leng = s.length();
   out.println("Length is: " +leng);
%>
```

\* while developing JSP based web application, while deploying and executing those web applications there is no necessity of adding any jar files to classpath because there is no need of compiling JSP programs from command prompt.

### Declaration

#### standard syntax

<%!

Decl. of instance variables,  
definitions of user defined methods,  
definitions of JspInit() & JspDestroy() life cycle methods

%>

#### xml syntax

<jsp: declaration>

—

</jsp: declaration>

\* The java code placed in declaration tag comes out side the \_JspService() of JSP equivalent servlet. So implicit objects are not visible in this declaration tag because all implicit objects are local variables in \_JspService() method.

Ex(1): <%! int a=10;  
int b=20; %>

\* Variables declared in declaration tag will come as the instance variables of the JSP equivalent Servlet class.

How to differentiate variables in the scriptlet when declaration tag variables and scriptlet tag variable are taken having same name?

(a) use 'this' key word (or) implicit object 'page' to differentiate declaration tag variable from the scriptlet tag variable.

<%! int a=10; %>

<% int a=20;  
out.println ("local variable (scriptlet) a value is:" +a);  
out.println ("instance variable (declaration) a value is:" +this.a); %>

Ex(2): keeping user defined java method definitions

<%! public int sum (int x, int y)  
{ return x+y;  
} %>

```

<% out.println("Result is:" + sum(10,20)); %>
* while working with xml syntax of declaration tag take care of < symbol
problem with the support of CDATA.

<jsp:declaration>
<! [CDATA[
    public int findBig(int a, int b)
    {
        if (a < b)
            return b;
        else
            return a;
    } //>
</jsp:declaration>

<% out.println("Big value is:" + findBig(10,20));%>

```

\* we can also use declaration tag to place JSP init, JSPDestroy() life cycle method definitions in our JSP program. By default the JSP equivalent servlet class does not provide these JSPInit(), JSPDestroy() life cycle methods.

### Ex 3:

```

<%! public void JSPInit()
{
    s.o.p("JSPInit(): can contain initialization");
} %>

<%! public void JSPDestroy()
{
    s.o.p("JSP Destroy(): can contain some uninitalization");
} %>
* JSP container calls JSPInit(), -JSPInit life cycle methods through init()
life cycle method when container instantiate JSP equivalent servlet class
* since JSP equivalent servlet automatically supplies -JSPInit() method the
programmer in JSP equivalent servlet the programmer should use only JSPInit()
method to place his initialization logic.

* Container calls JSPDestroy(), -JSPDestroy() methods through destroy() life
cycle method when Container is about to destroy the object of JSP
equivalent servlet class.

* since JSP equivalent servlet automatically supply -JSPDestroy() method the
programmer should use JSPDestroy() in his JSP program to place on
initialization logic.

```

NOTE: only Tomcat 6.0 onwards -`_jspInit()`, `_jspDestroy()` methods are introduced in earlier versions of tomcat and in other servers -`_jspInit()`, `_jspDestroy()` methods are not there.  
The overall conclusion is -`_jspInit()`, `_jspDestroy()` methods are useless methods for programmer.

\* we can not define -`_jspService(..)` method in declaration tag of JSP program because jsp equivalent servlet automatically gives that method so our method becomes duplicate method to it and java does not support to have duplicate methods in a java class.

### In JSP Program

```
<%! public void _jspService(HttpServletRequest req, HttpServletResponse res)
{
}%>
```

In valid code

### Expression tag :

#### standard syntax

```
<% = java expression %>
```

#### xml syntax

```
<jsp:expression>
    java expression
</jsp:expression>
```

- \* This tag evaluates given expression and writes generated result to browser window.
- \* Code placed in expression tag goes to -`_jspService(..)` of jsp equivalent servlet so implicit objects are visible in expression tag.
- \* Expressions are nothing but arithmetic operations, logical operations and method calls.

```
<% int a=10;
    int b=20;
    out.println(a+b); %>
```

(equals)

```
<% a=10;
    int b=20; %>
<%= a+b%>
```

Evaluates a+b expression and writes the result to browser window.

\* we can also use the expression tag the values of the variables.

```
<% int a=10;  
    int b=20; %>  
A value is: <%= a %>  
B value is: <%= b %> Result is:  
<%= a+b %>
```

\* we can use expression tag to call java methods (both pre defined and user defined).

```
<%! public int sum(int x, int y)  
{  
    return x+y;  
}%>
```

The result is: <%= sum(10,20) %> <br>

```
<% String s="Hello how r u"; %>
```

The substring of <%= s %> is: <%= s.substring(0,5) %>

\* If expression tag is used to instantiate a java class then the default data of the object will be written to browser window as response.

Ex: Date and Time is: <%= new java.util.Date() %>

### Example on xml based expression tag

```
<jsp:scriptlet>  
    int a=10;  
    int b=20;  
</jsp:scriptlet>  
Result is: <jsp:expression> a+b </jsp:expression>
```

\* By using xml syntax of expression tag we can not pass expressions that are using "<" symbol. Even though CDATA option is used

```
<jsp:scriptlet>  
    int a=10;  
    int b=20;  
</jsp:scriptlet>                                Invalid code  
Result is: <jsp:expression>  
    <! [CDATA [a<b]]>  
</jsp:expression>
```

\* The above problem can be solved with standard syntax of expression tag.

Ex: <jsp:scriptlet>  
int a=10;  
int b=20;  
</jsp:scriptlet>

Result is: <% = a < b %>

\* It is always recommended to use scripting elements of JSP by following standard syntax.

\* The JSP expression tag can evaluate only one expression at a time.

It can not be used to evaluate multiple expressions as shown below.

<% int a=10;  
int b=20; %>

Results are: <% = a+b, b+a, a+a %>  
wrong statement

\* we can not write one scripting tag of JSP inside another scripting tag  
that means we must use these tags as independent tags.

<% int a=10;  
int b=20;  
<% = a+b %> — Invalid statement  
%>

Procedure to develop JSP based web application by using netbeans IDE

Step 1: Create web project having name JspApp2

File menu → new project → Java Web → web application → next →

Project name: JspApp2 → next → server: GlassFish 2.1 → next → finish

Step 2: Add JSP program to the web pages folder of JspApp2 web application

Right click on web pages folder → new → JSP → JSP file name: First →

Finish.

First.jsp

<%!  
public String getWishMsg(String uname)  
{  
 // Current date and time  
 java.util.Calendar d = java.util.Calendar.getInstance();  
 // Get current hour of the day  
 int h = d.get(java.util.Calendar.HOUR\_OF\_DAY);  
 // Generate wish msg  
 if (h <= 12)

```

        return "Good morning" +uname
    else if ( h <= 16 )
        return "Good afternoon" +uname
    else
        return "Good night" +uname
    } %>
Date and time is: <% = new java.util.Date() %> <br>

<% String uname = "roja";
%> <br> the wish msg is: <% = getwishmsg(uname) %>

```

Step(3): Run the project

Right click on Project → Run.

Re writing the above First.jsp with xml syntax

```

<jsp:declaration>
<!CDATA[
Public String getwishmsg (String uname)
{
    ==
}
</jsp:declaration>

Date and time is: <jsp:expression>new java.util.Date()</jsp:expression>
<jsp:scriptlet> String uname="roja"; </jsp:scriptlet> <br>
The wish msg is: <jsp:expression>getwishmsg(uname)</jsp:expression>

```

+ we can install tomcat server in two ways

(i) through ~~setup~~ file (to installer) (.war)

(ii) through zip file

+ The Tomcat server that installed through zip file can only configured with netbeans IDE

Procedure to Configure external tomcat with netbeans IDE

Step(1): Install tomcat that is there in zip file mode (apache-tomcat-6.0.33-binary.zip file) (Extract this file to a folder)

Step(2): Configure the above tomcat server with netbeans IDE.

Tools → servers → Add Server → Tomcat 6.0 → next →

server location: <Tomcat-home>\apache-tomcat-6.0.33

The folder that comes after zip file extraction.

----- configuration folder  
catalina Base: <Tomcat-home>\apache-tomcat-6.0.33  
User name : admin  
Password : admin → finish → close

Step 3: Link Tomcat server to web project of Netbeans IDE

Right click on web project → properties → run → server : Tomcat 6.0 → OK.

Step 4: Run the web project

Procedure to configure Adiu JavaBatch Domain server of WebLogic 10.3 with Netbeans IDE

Tools → servers → Add Server → Oracle WebLogic Server → next

Server location: D:\oracel\middleware\wlserver\_10.3 → next

Local instances: Admin server (localhost:7001)

username: javaboss

password: javaboss1 → Finish → close

\* Link WebLogic server with your web project

same as above

\* Run the web project.

### Implicit objects

request : javax.servlet.HttpServletRequest → request scope

response : javax.servlet.HttpServletResponse → response scope

out : java.servlet.jsp.JspWriter (AC) → page scope

session : javax.servlet.http.HttpSession (I) → session scope

page : this (currently invoking JSP equivalent servlet class obj ref) → page scope

PageContext : javax.servlet.jsp.PageContext (C) → page scope

application : javax.servlet.ServletContext (I) → web application scope

config : javax.servlet.ServletConfig (I) → page scope

exception : java.lang.Throwable (C) → page scope

All the above given 9 implicit objects of JSP are the objects of servlet container and JSP container supplied Java classes implementing the above given interfaces or extending from the base interfaces of Servlet, JSP APIs given by Sun Microsystems.

\* JSP programmers never worries and remembers the class names to of these implicit objects because they will be changed based on the server we utilize.

Ex: The implicit object out is not the object of javax.servlet.jsp.JspWriter class. It is the object of underlying JspContainer supplied java class that extends from javax.servlet.jsp.JspWriter class.

In Tomcat server that class name is : org.apache.jasper.runtime.JspWriterImpl

In weblogic server that class name is : weblogic.servlet.jsp.JspWriterImpl.

\* In JSP program we can use 3 types of comments to comment the code

(1) Scripting Comments (Java comments) (// and /\* .... \*/)

given to comment the Java code of scripting elements (<% ... %>, <%! ... %>, <%= ... %>)

(2) HTML Comments / Template Text Comments (<!-- ..... -->)

given to comment the HTML code and template text code of JSP program.

(3) Hidden Comments / JSP Comments (<%-- ..... --%>)

given to comment the JSP tags of JSP programs

\* Every Compiler or interpreter generates white spaces for commented code so we get the feeling the commented code is not participating in execution.

+ JSP page compiler recognizes JSP comments and generates white spaces in the source code of JSP equivalent Servlet program.

\* Java comments will be recognized by javac Compiler and keeps white spaces in .class file.

+ HTML comments will be recognized by HTML interpreter.

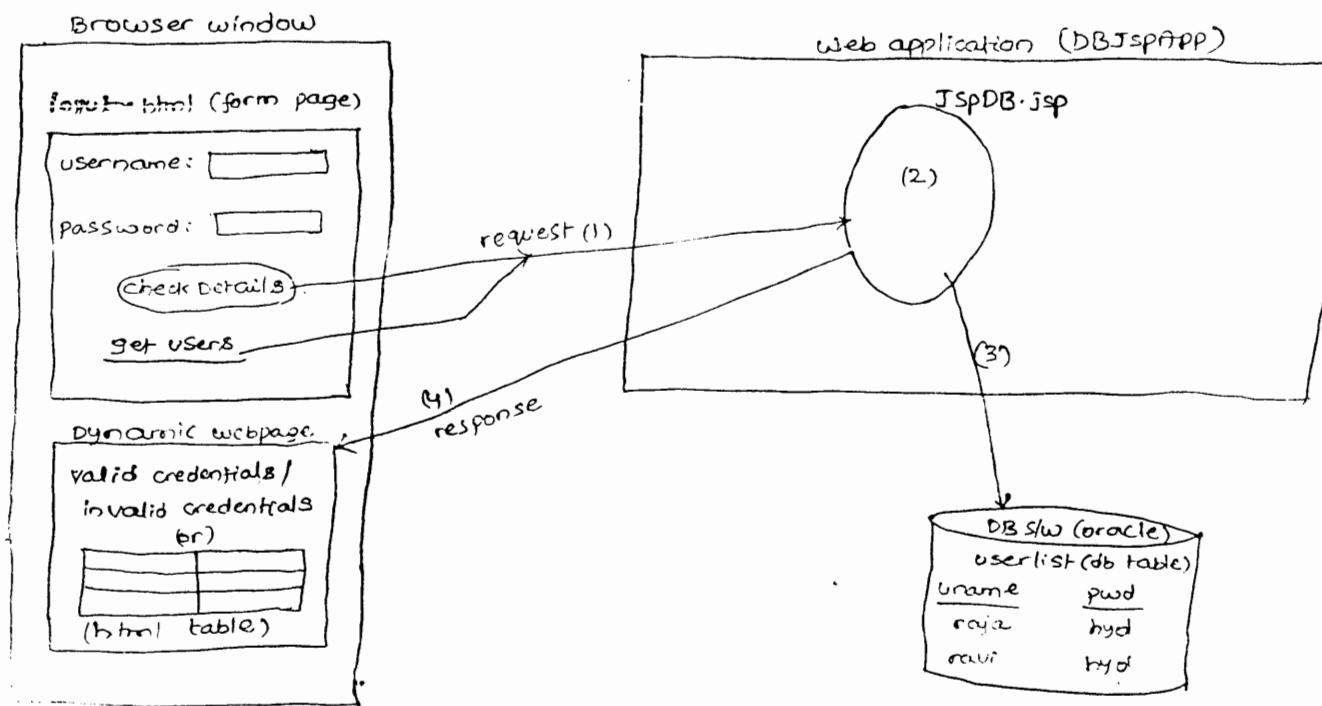
+ the JSP program related JSP comments are not visible in any phase of JSP programs execution so they are technically called as hidden comments.

\* we can comment HTML code of JSP program with JSP comments but not recommended. we can comment JSP tags with HTML comments but it is also not recommended.

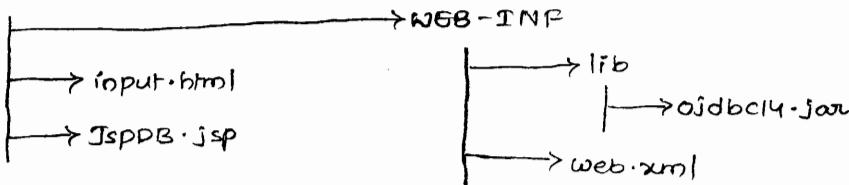
comment type	visibility			
	In src code of Jsp equivalent servlet	In Compiled code of Jsp equivalent Servlet	In the html code that goes to browser window	in the output/response
JSP Comment <%-- ..... --%>	no	no	no	no
html Comment <!-- ... -->	yes	yes	yes	no
bt scripting Comment (// or /*--- */)	yes	no	no	no

- \* The HTML program related web page can have form or hyperlink to generate request to JSP or servlet program. This provides GUI ness to end user to communicate with servlet/JSP program from browser window.
- \* For HTML to JSP Communication you can use the JSP file name or its URL pattern as action url of form page (as action attribute value of form tag) or as href of anchor tag.
- \* For JSP to DB Communication place JDBC code in JSP program.

Example application on html to JSP and JSP to DB S/w communication



## DBJspAPP



\* The code that comes to `_jspService(--)` method of `JSP` equivalent servlet will be automatically handling the exceptions because that `_jspService(--)` method code will be there surrounded with try and catch blocks.

\* The code placed in declaration tag of JSP program must handle the exception explicitly whereas the code placed in scriptlet, expression tag automatically handles the exception.

### Source code of the above diagram based application

#### input.html

```
<form action="JspDB.jsp" method="get">  
    Login Username: <input type="text" name="tname"/><br>  
    Login password : <input type="password" name="tpwd"/><br>  
    <input type="submit" value="check details" name="s1"/>  
</form> <br><br><br>  
<a href="JspDB.jsp? s1=link" > Get All user details </a>
```

#### JspDB.jsp

```
<%@ page import = "java.sql.*"; %> //taken to import the java packages  
<%!  
    Connection con;  
    PreparedStatement ps1, ps2;  
    public void JspInit()  
    {  
        try  
        {  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:  
                                              sathya", "scott", "tiger");  
            ps1 = con.prepareStatement("Select count(*) from userlist where  
                                     uname=? and Pwd=?");
```

```

PS2 = Con.prepareStatement("Select * from userlist");

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

//ispInit()

%>

<% //read form data from form.page
String user = request.getParameter("tname");
String pwd = request.getParameter("tpwd"); //read additional req param value
String param = request.getParameter("si");

if (param.equals("link")) //when hyperlink is clicked
{
    //write jdbc code to get all users details

    ResultSet rs = PS2.executeQuery();
    out.println("<Table>");
    while(rs.next())
    {
        out.println("<tr>");
        out.println("<td>" + rs.getString(1) + "</td>");
        out.println("<td>" + rs.getString(2) + "</td>");
        out.println("</tr>");
    }
    out.println("</table>");
    rs.close();
}

else //when submit button is clicked
{
    //write jdbc code for authentication

    //read form data
    String user = request.getParameter("tname");
    String pass = request.getParameter("tpwd");
    //set form data as param values of query

    PSI.setString(1, user);
    PSI.setString(2, pass);
    //execute the query
    ResultSet rs = PSI.executeQuery();
}

```

rs (ResultSet obj)	
raja	hyd
ravi	hyd

BFR      ALR  
 logic to display  
 ResultSet obj data  
 as html content

rs (ResultSet obj)	
.	

```

int count = 0;
if (rs.next())
    count = rs.getInt(1);
if (count == 0)
    out.println("<b><i><font color=red>Invalid credentials </font></i></b>");
else
    out.println("<b><i><font color=red>Valid credentials </font></i></b>");
}
%>
<% ! public void JspDestroy()
{
    try
    {
        PS1.close();
        PS2.close();
        con.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
//JspDestroy()

```

### web.xml

<web-app>

+ keep ojdbc14.jar file in WEB-INF\lib folder.

+ Deploy the DBJspApp in Tomcat server.

+ use the following request URL to test the application.

http://localhost:2020/DBJspApp/input.html

### Userlist (DB table)

<u>uname</u>	<u>pwd</u>
raja	hyd
ravi	hyd

Select \* from userlist where uname='raja' and pwd='hyd'

Count(\*)

1 → Valid credentials

Select \* from userlist where uname='raja' and pwd='hyd'

Count(\*)

0 → Invalid credentials

## Page Directive Tag

\* This tag is given to provide global information to JSP program like buffer size, package import statements, Content type and etc...

### standard syntax

<%@ page attributes %>

### xml syntax

<jsp:directive.page attributes />

### attributes

language = "java" default → java // Java is the default and only one language that you can pass here as a value.

\* The above attribute specifies the language that we need to use to write code in scripting tags. (only Java is allowed).

import = "java.util.\*, java.net.\*, ..." // default : java.lang.\*

\* useful to import the Java packages that are required for Java application JSP program.

### extends = "class name"

\* Not recommended to use, no default value.

\* Generally the JSP equivalent servlet class extends from one or other JSP container supplied predefined Java class. If you want to see that JSP equivalent servlet class extending from your own or your choice Java class then use this extends attribute. Since programmer can not develop that full fledged Java class should act as super class of JSP equivalent servlet class so this extends attribute is not recommended to use.

contentType = "text/html" // resp content type, // no default value

\* this is useful to specify content type of the response that should be generated by JSP program.

Ex: <%@ page language = "java" import = "java.net.\*, java.sql.\*" contentType = "text/html" %>

bufSize = "18kb" or more // default → 8kb

- specifies the size of the buffer at server side.

\* Even though the code of JSP program is executing part by part, because of underlying CPU algorithms (like round robin algorithm) the server preserves output of the executed statements in buffer of the JSP program wherein until last statement of JSP program gets executed.

Ex: `<%@ page buffer = "10kb" %>`  
`<b>Hello</b>` ↳ The buffer size of JSP program at server side

Ex: `<%@ page buffer = "none" %>`  
↳ makes the JSP program running without buffer.

NOTE: Every servlet program also contains one default server side buffer. But there is no default size for this buffer.

\* The server side buffers servlet/JSP program is no way related with client side (browser window) buffer.

\* Buffer is a temporary memory that can hold data for temporary period.

autoFlush = "true" (false) //default → true

\* It cleans the buffer of JSP program and sends to browser window as web page

Ex: `<%@ page buffer = "8kb" autoFlush = "true" %>`

session = "true" (false) //default → true

\* Enables JSP equivalent servlet create or not to create implicit object session

Ex(1): `<%@ page session = "false" %>`

↳ implicit object session will not be created.

Ex(2): `<%@ page session = "true" %>`

↳ implicit object session will be created.

+ See when we don't enable session tracking we don't need session object in JSP program. To disable that session object creation use above statements.

errorPage = "url" //no default

iSErrorPage = "false" (true) //default → false

} Required to configure error pages for JSP program to handle the exceptions raised in JSP program.

iS ThreadSafe = "true" (false) //default → true

\* Useful to enable or disable thread safety on JSP equivalent servlet

Ex: `<%@ page iS ThreadSafe = "false" %>`

make JSP equivalent servlet as ThreadSafe servlet by implementing javax.servlet.SingleThreadModel (i)

info = "String"

\* The short desc about JSP page - no default value.

Ex: <%@ page info="basic program" %>

iB<sub>E</sub>LIgnored = "true" (false) //default → false

\* To perform arithmetic and logical operations in JSP program without using java code, we can use EL (Expression Language).

\* The above attribute can make page compiler to ignore or recognize that EL.

Rules to remember while working with page directive tag (<%@ page %>)

(1) Page directive tag name and attribute names are case sensitive.

(2) Unknown attributes will not be recognized.

(3) Except import attribute no other attribute can have multiple values separated with ';' symbol

<%@ page import="java.net.\* , java.awt.\*" buffer="8kb,10kb" %>

It is allowed

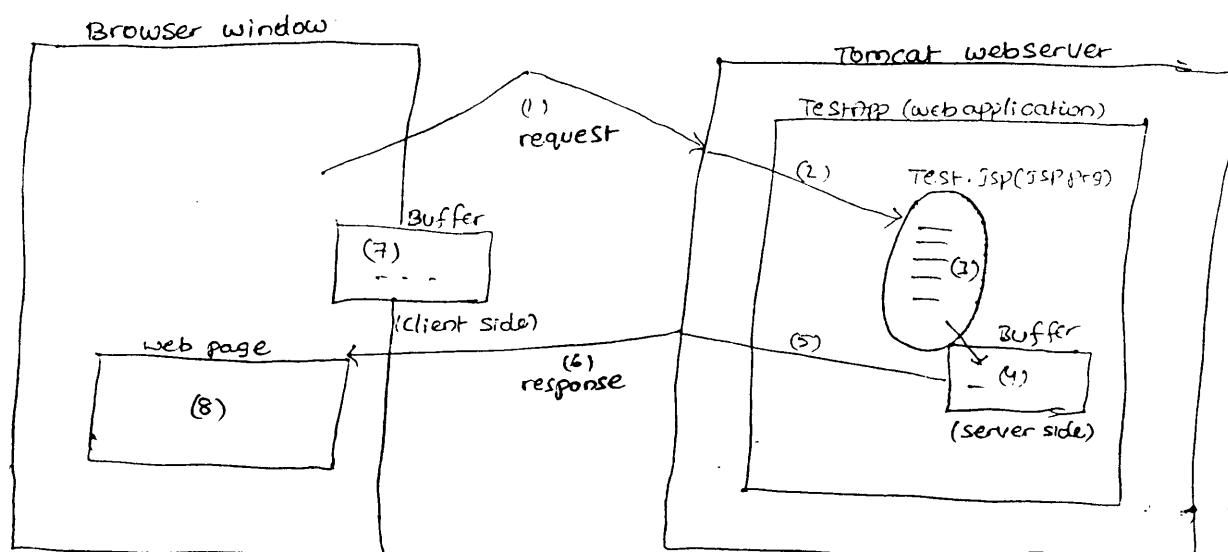
It is not allowed

(4) Except import attribute no other attribute can not occur for multiple times with different values either in the same page directive tag or in the multiple page directive tags of same Jsp program.

<%@ page import="java.net.\*" import="java.net.\*" buffer="8kb" buffer="10kb" %>

allowed

In valid



\* The JSP program buffer size is taken as 1kb but that JSP program is generating more than 1KB output then what happens?

(a) The JSP program dynamically expands its size as needed.

NOTE: Even though buffer = none is taken the JSP program is capable of expanding buffer size as needed <sup>at</sup> runtime so we can enable autoFlush = "true" even though buffer is none.

Ex: <%@ page buffer="none" autoFlush="true" %>

what is the difference between PrintWriter() and JspWriter()?

\* The process of storing result/output in a buffer before delivering to the original destination is technically called as buffering operation.

\* PrintWriter can not perform buffering while sending its output to browser window whereas JspWriter can perform buffering.

\* When JSP program is taken with out buffer then JspWriter internally uses PrintWriter to write the output to browser window.

\* JspWriter is the type of the implicit object called out in JSP Program.

\* PrintWriter is useful in servlet Program.

\* PrintWriter belongs to java.io package and JspWriter belongs to javax.servlet.jsp package

— . —

\* Servlet API is given as

javax.servlet package, javax.servlet.http package

Latest version of servlet API is 3.0 (supports Annotations based programming)

running version of servlet API is 2.4/2.5

\* JSP API is given as

javax.servlet.jsp package

javax.servlet.jsp.el package

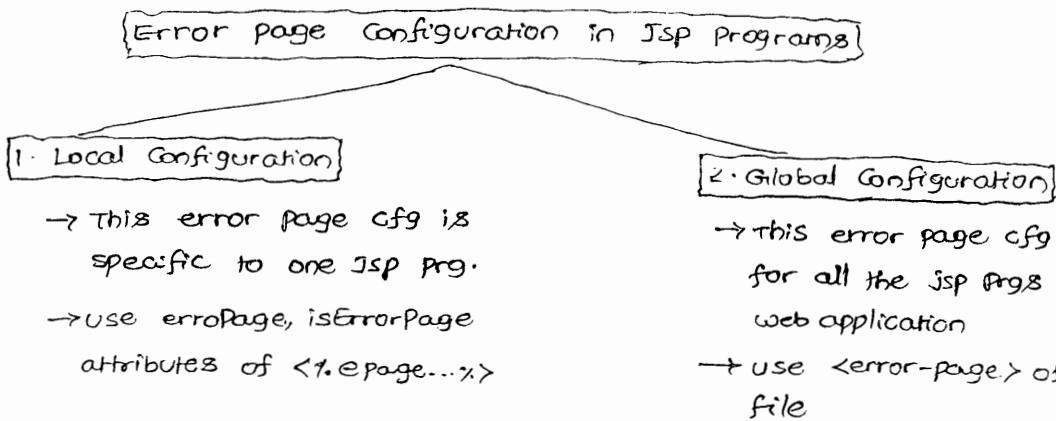
javax.servlet.jsp.tagext package

Latest version of JSP API is 2.1

running version of JSP API is 2.0

\* JSP equivalent servlet automatically handles the exceptions but displays the exceptions related error messages quite ugly on browser window when exception is raised. Since these are technical messages the non-technical end users can not understand them. To overcome this problem configure error pages for JSP program.

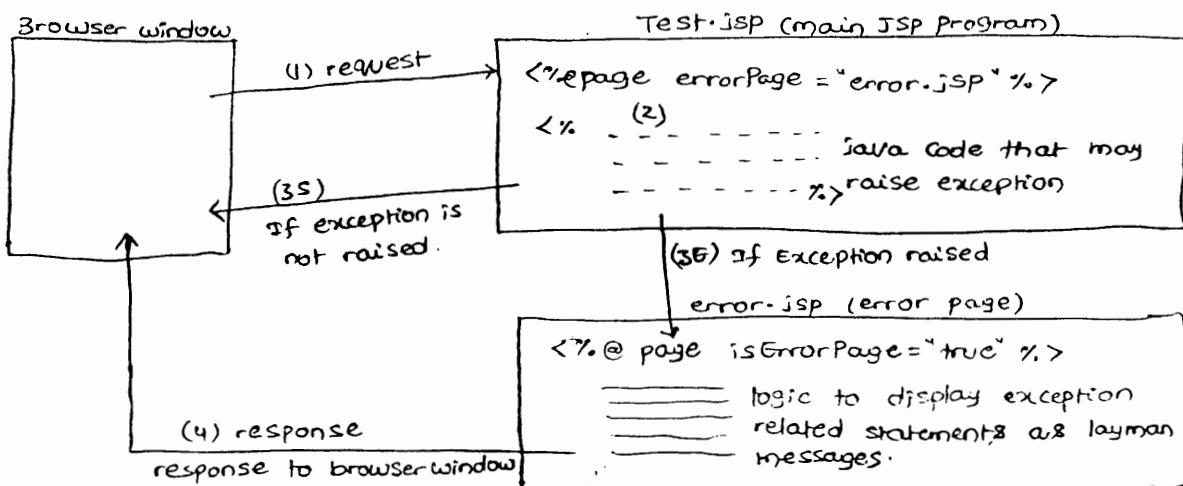
- \* Error page is a JSP or HTML program that are capable of executing only when exception is raised in other JSP programs. These are also useful to display the exception related messages non technical messages guiding the end users.
- \* Always develop webapplications by keeping non-technical end users in mind. For that error pages configuration is important.



In servlet program we take support of `rd.forward(...)` for Error servlet/Error page configuration.

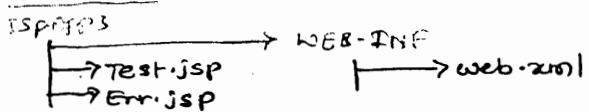
- \* The Errorpage Configuration done in `web.xml` file for JSP programs (global configuration) will not work for servlet programs of web application.

### Local Error page Configuration



- `isErrorPage="true"` attribute of `page directive tag` can make a JSP program of web application as error page.
- The implicit object `Exception` is visible only in that JSP program that acts as error page and we can use this object to know the details of exceptions that are raised in main JSP programs.

### Example application



### Test.jsp

```
<%@ page errorPage = "Err.jsp" %>
<b> from Test.jsp </b><br>
<% int x=Integer.parseInt("a123");
    out.println("x value is :" +x); %>
```

### Err.jsp

```
<%@ page isErrorPage = "true" %>
<b> from err.jsp </b><br>
<font color = red> Internal problem </font> <br>
The exception that is raised is : <%= exception.toString() %>
                                         Implicit object
```

### web.xml

```
<web-app>
```

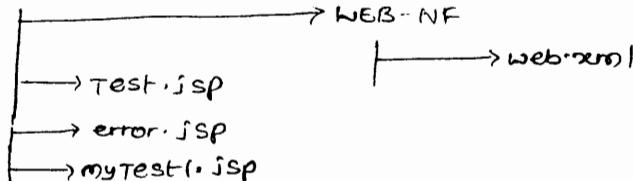
Request URL to test the application

<http://localhost:2020/StrutsApp3/Test.jsp>

\* The above exception handling configuration also allows to take .html file as error page but we can not choose to work with implicit object exception in that error page.

### Example application on global error page configuration

#### JspApp



### Test.jsp

```
<b> from Test.jsp </b><br>
<% int x = Integer.parseInt("a123");
    out.println("x value is :" +x); %>
```

### MyTest.jsp

```
<b> from myTest.jsp </b><br>
<% Class.forName('java.lang.System');
    %>
```

### Error.jsp

\* Same as the above err.jsp

### web.xml :

```
<web-app>
  <error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/err.jsp</location>
  </error-page>
</web-app>
```

NOTE: Because of the above configuration done in web.xml file for any exception raised in any JSP program of web application the control goes to error.jsp page.

\* Global error page configuration also allows to take .html program as error page.

\* If both local error page , global error page configurations are done in same page to handle same exception pointing to two different error pages can you tell me which settings will be effected ?

(a) The configurations done in local error page configuration will be effected.

---

### Directive include tag

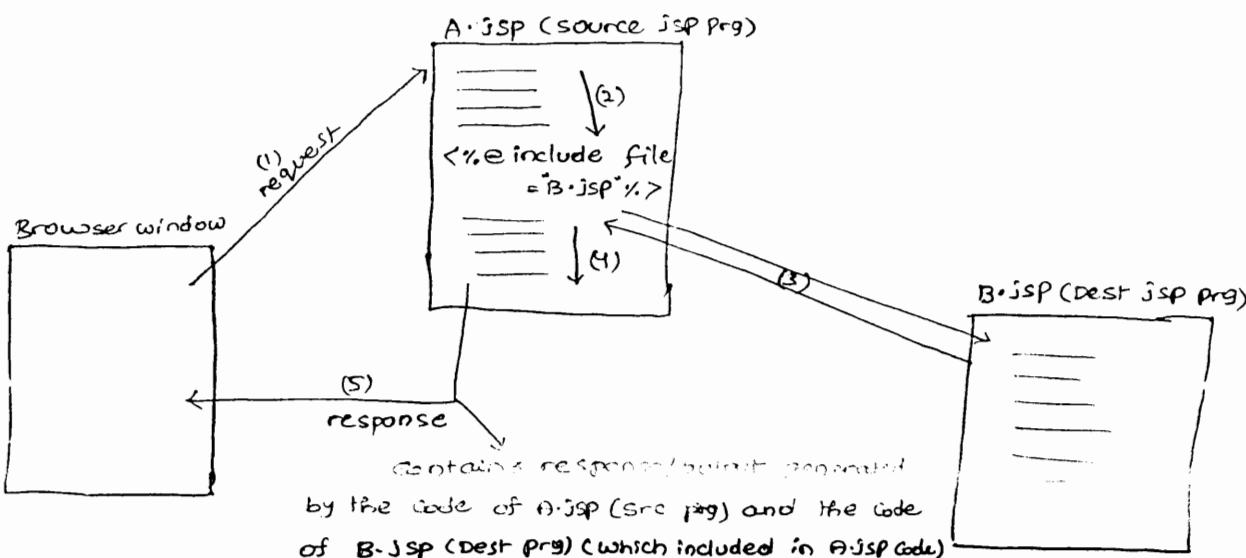
#### standard syntax :

```
<%@include file="dest url"%>
```

#### xml syntax

```
<jsp:directive.include file="dest url"%>
```

\* This tag is given to include the code or content of destination web resource program to the source code of JSP equivalent servlet belonging to the source JSP program.

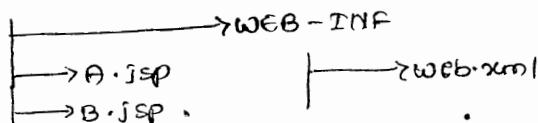


here B.jsp will not be executed separately but its code will be included to the code of A.jsp and executes along with A.jsp.  
(refer step (3) of diagram).

\* Directive Include (<%@ include ... %>) does not perform output inclusion but it performs code/ content inclusion.

Example:

JspApp5



A.jsp

```
<b> from A.jsp </b> <br>
<%@ include file="B.jsp" %>
<b> from end of A.jsp </b>
```

B.jsp

```
<b> from B.jsp </b> <br>
<%; = new java.util.Date(); %>
```

web.xml

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

Request URL to test the application

```
http://localhost:2020/JspApp5/A.jsp
```

\* when the above A.jsp is requested

- JSP page Compiler generates JSP equivalent servlet for A.jsp
- Because of <%@ include file="B.jsp" %> the code of B.jsp will be included to A.jsp related JSP equivalent servlet program
- The Java Compiler compiles JSP equivalent servlet source code of A.jsp and executes that code then sends response to browser window.

\* In the above execution B.jsp program will not be executed separately so its JSP equivalent servlet will not be generated separately.

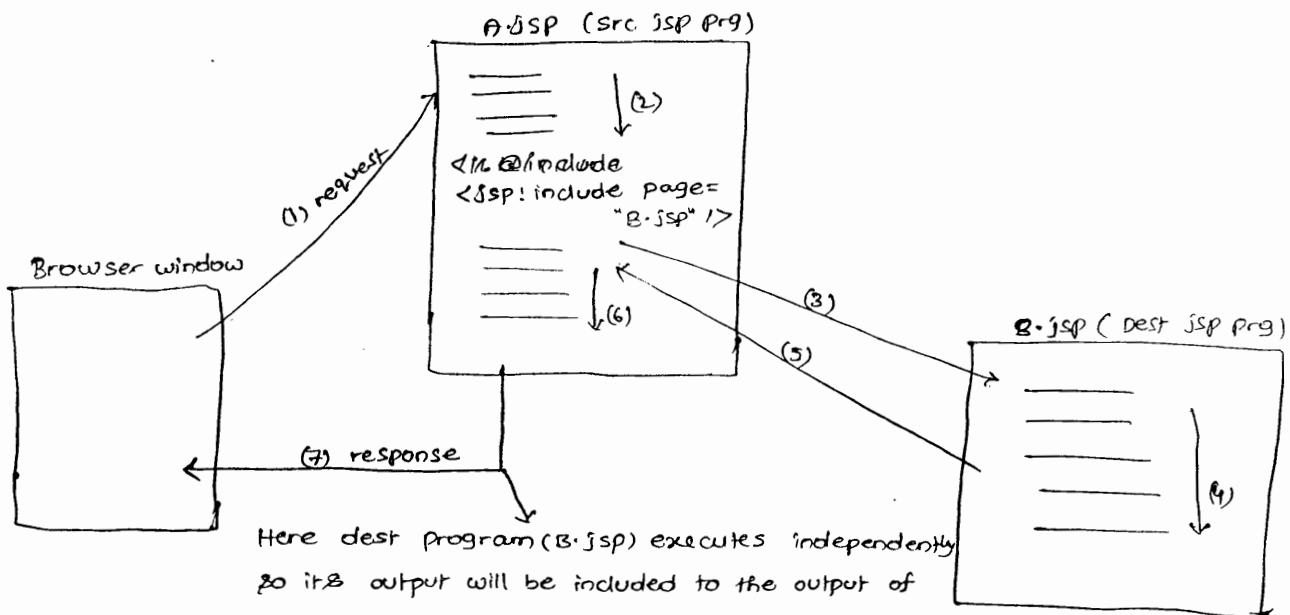
Action Include (<jsp:include>)

syntax

```
<jsp:include page="dest url"/>
```

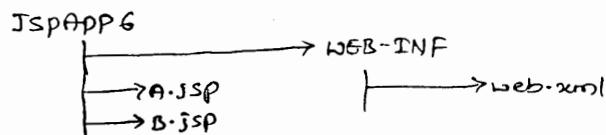
NOTE: All standard Action tags of JSP like <jsp:include> are having only XML syntax and there is no standard syntax for them.

\* <jsp:include> is given to include the output of destination web resource program to the output of to the source JSP program.



Here dest program (B.jsp) executes independently so its output will be included to the output of src program (A.jsp) dynamically. The response that comes to browser window contains the output of A.jsp (src prg) and B.jsp (dest prg) programs.

### Example application



#### A.jsp

```

<b> from A.jsp </b><br>
<jsp: include page="B.jsp"/>
<b> from end of A.jsp </b>
  
```

#### B.jsp

same as B.jsp in JspApp5

web.xml

URL to test

<http://localhost:2020/JspApp6/A.jsp>

\* <jsp:include> performs output inclusion but not the code inclusion.

\* <jsp:include> is tag based alternate for rd::include(--) method of servlet programming.

\* when the above A.jsp is requested two separate JSP equivalent servlets will be generated for both A.jsp, B.jsp and the real JSP equivalent servlet of A.jsp internally uses some rd::include(--) method to include the output of B.jsp.

\* The action tags of JSP can be used only with XML syntax compare to other tags of JSP (other tags can be having both XML and standard syntax).

\* Standard Action tag of JSP are given to perform more dynamic operations at runtime/request processing phase operation.

What is the difference between "Include Directive" and "Action include tag"?

#### Include Directive

`<%@ include ....%>`

#### Action Include

`<jsp:include>`

\* Given to perform code inclusion.

\* Performs code inclusion at translation phase so this work is called as compile time binding or static binding.

\* If destination program is JSP the separate JSP equivalent servlet will not be generated for that program.

\* Can not include code and output of servlet program.

\* Suitable to take static web resource programs as destination programs.  
(like HTML programs)

\* Does not use rd.include(--) internally.

`<%@ include file="B.jsp"%>`

\* Given to perform output inclusion.

\* Performs output inclusion at run time/request processing phase so this work is called as dynamic/runtime binding.

\* Will be generated.

\* Can include (includes the output).

\* Suitable to take dynamic web resource programs as destination programs  
(like servlet program, JSP program)

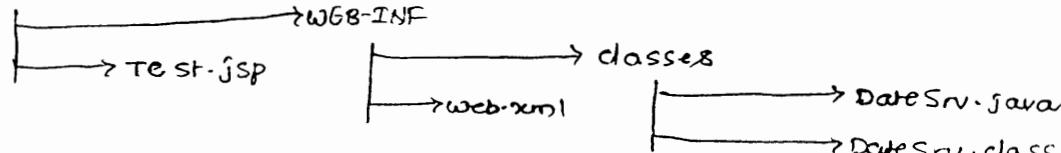
\* Uses rd.include(--) internally.

`<jsp:include page="B.jsp"/>`

#### Example application on JSP program

##### Program

JspApp7



Request URL

`http://localhost:2020/JspApp7/Test.jsp`

##### Working including the output of servlet

### Date Srv.java

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

public class DateSrv extends HttpServlet
{
    public void service(HttpServletRequest req, HttpServletResponse res) throws
        ServletException, IOException
    {
        // general settings
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        // write b.logic
        java.util.Date d = new java.util.Date();
        pw.println(d.toString());
    }
}
```

// class

### web.xml

Configure DateSrv Servlet program with test1 url pattern.

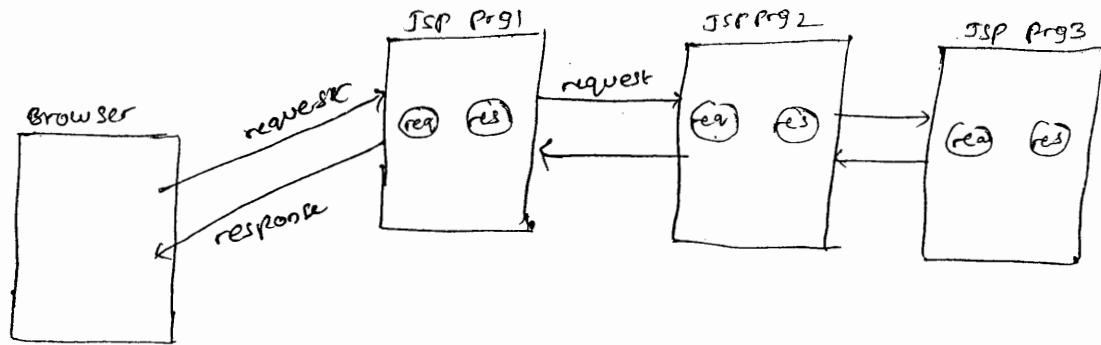
### Test.jsp

<b> from begining of Test.jsp </b> <br>  
Date and Time is <jsp:include page="test1"/> <br>  
<br> <i> end of Test.jsp </i> we can not replace this tag with include  
directive tag

\* while working with directive include and action include don't commit  
the response in destination web resource program by using out.close()  
(or pw.close() method) because it does not allow to add further  
output of source Jsp program to the response.

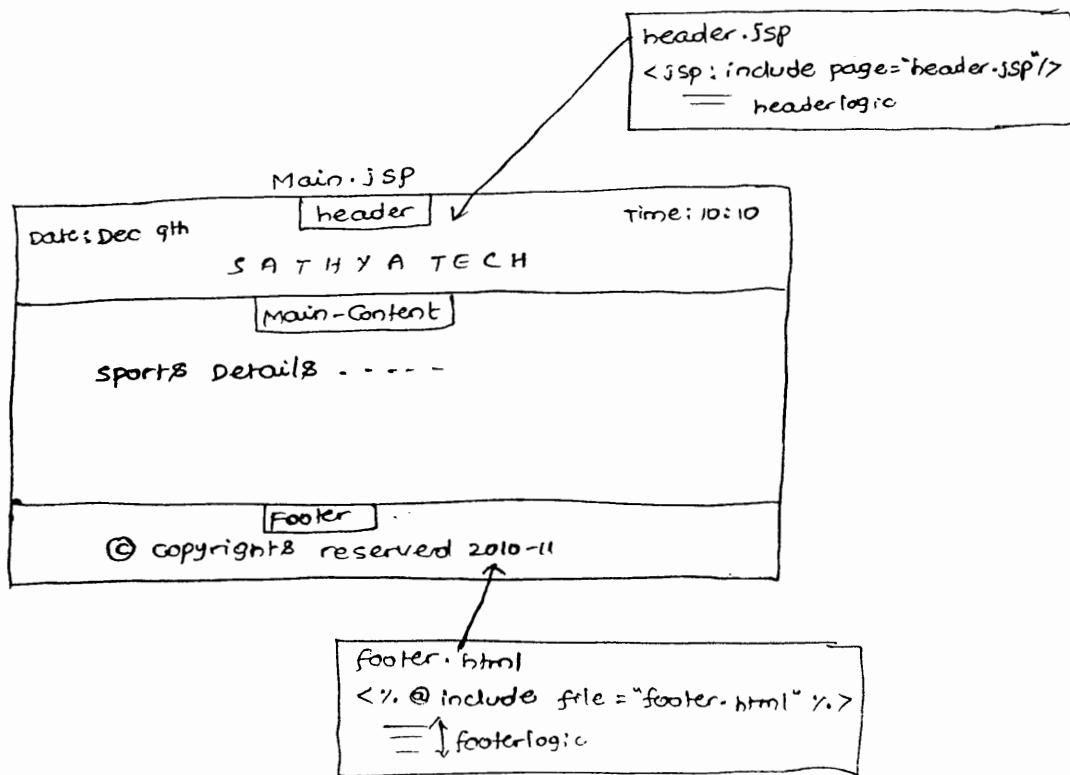
\* taking a request from a browser window and process the request by  
using multiple JSP programs as chaining. for this we can use <jsp:include>  
tag or <jsp:forward> tag.

\* all the JSP programs which participate in a JSP chaining will use  
same request and response objects.



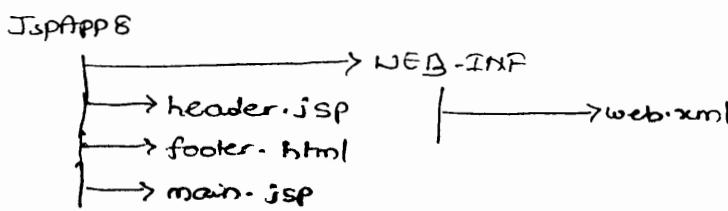
Java programs participating in JSP chaining, all the steps JSP programs of JSP chaining will use same request and response objects.

- \* If the content to include is static content use directive include tag. If the content to include is dynamic content use action include tag.



- + In one JSP program we can see the utilization of both directive include, action include tags for multiple times.

Example application based on the above diagram



### header.jsp

```
<%@ page import = "java.util.*" %>  
<% Calendar cl = Calendar.getInstance();  
    int date = cl.get(Calendar.DAY_OF_MONTH);  
    int month = cl.get(Calendar.MONTH); %>  
  
<%= date %>/<%=month %>    &nbsp;&nbsp;&nbsp;&nbsp;<%=cl.getTime()%>  
<br><br>  
<marquee> <font color = red size = 5> S A T H Y A T E C H </font> </marquee>  
<br>  
<hr>
```

### footer.html

```
<hr>  
<br><br> <center>  
    &copy; copy rights reserved 2010-11 </center>
```

### Main.jsp

```
<jsp:include page = "header.jsp"/>  
<br><br>  
Sports news of the day <%=new java.util.Date()%>;  
    <font size = 2> Sehwag has scored 219 runs in an innings of one day  
        cricket match beating Sachin's record of 200. </font> <br><br><br>  
<%@ include file = "footer.html" %>
```

### web.xml

```
<web-app>
```

Request URL to test the application

http://localhost:2020/<sup>Jsp</sup>guruApp8/main.jsp

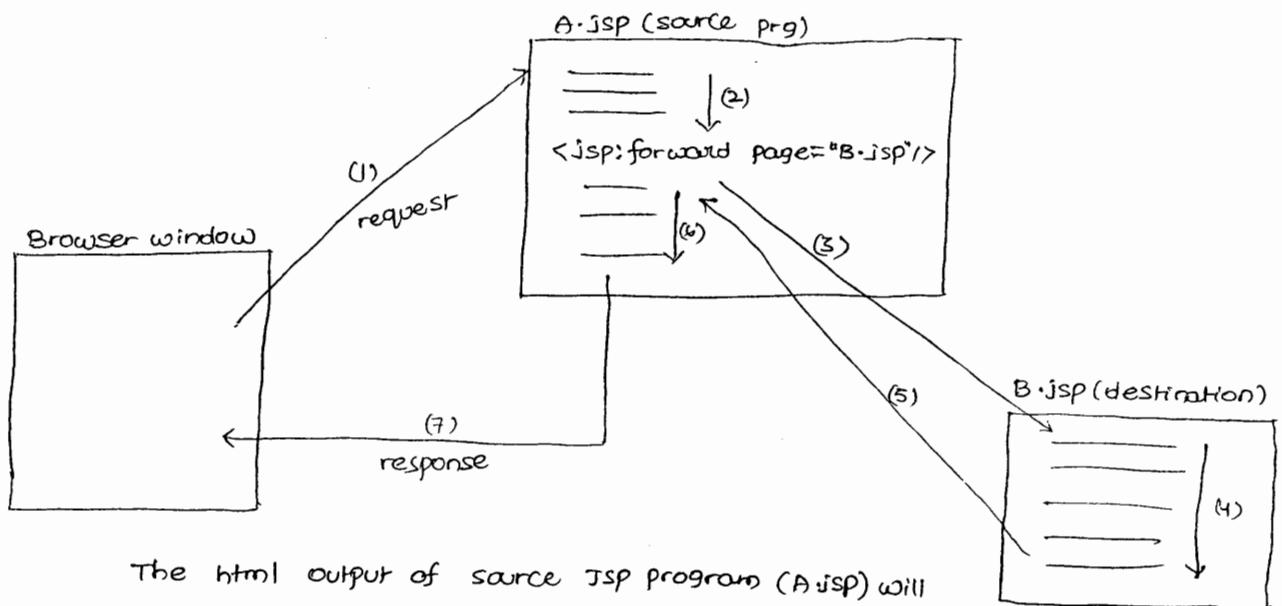
\* For summary table that talks about all JSP tags refer page nos 112 & 113.

### <jsp:forward>

#### Syntax

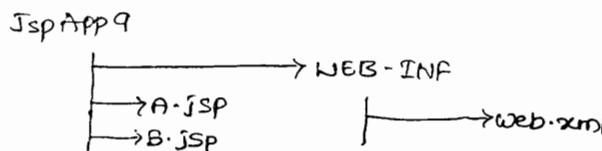
```
<jsp:forward page = "dsturl"/>
```

\* Given to perform forwarding request mode of JSP chaining. It internally uses  
rd.forward(---) method.



The html output of source JSP program (A.jsp) will be discarded and only the html output of (B.jsp) (destination prg) goes to browser window as response through A.jsp (src jsp prg). Here two separate and independent equivalent servlet programs will be generated for A.jsp, B.jsp programs.

### Example application



### A.jsp

```

<b> start of A.jsp </b> <br>
<jsp:forward page="B.jsp"/>
<b> end of A.jsp </b>

```

### B.jsp

```

<br><%=new java.util.Date()%> <br>
<i> from B.jsp </i>

```

\* When source JSP program is interacting with destination JSP program either by using `<jsp:forward>` or `<jsp:include>` then source JSP program can send additional request parameter values to destination JSP program by using `<jsp:param>` tag.

↓  
This tag is allowed to use only as sub tag of `<jsp:include>`, `<jsp:forward>` tags.

### Example

#### In A.jsp of JspApp9

<b> start of A.jsp </b> <br>

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

<jsp:param name="bookname" value="CRJ"/>  
<jsp:param name="price" value="300"/> } Additional request parameters  
name, values

</jsp:forward>

<b> end of JSP </b>

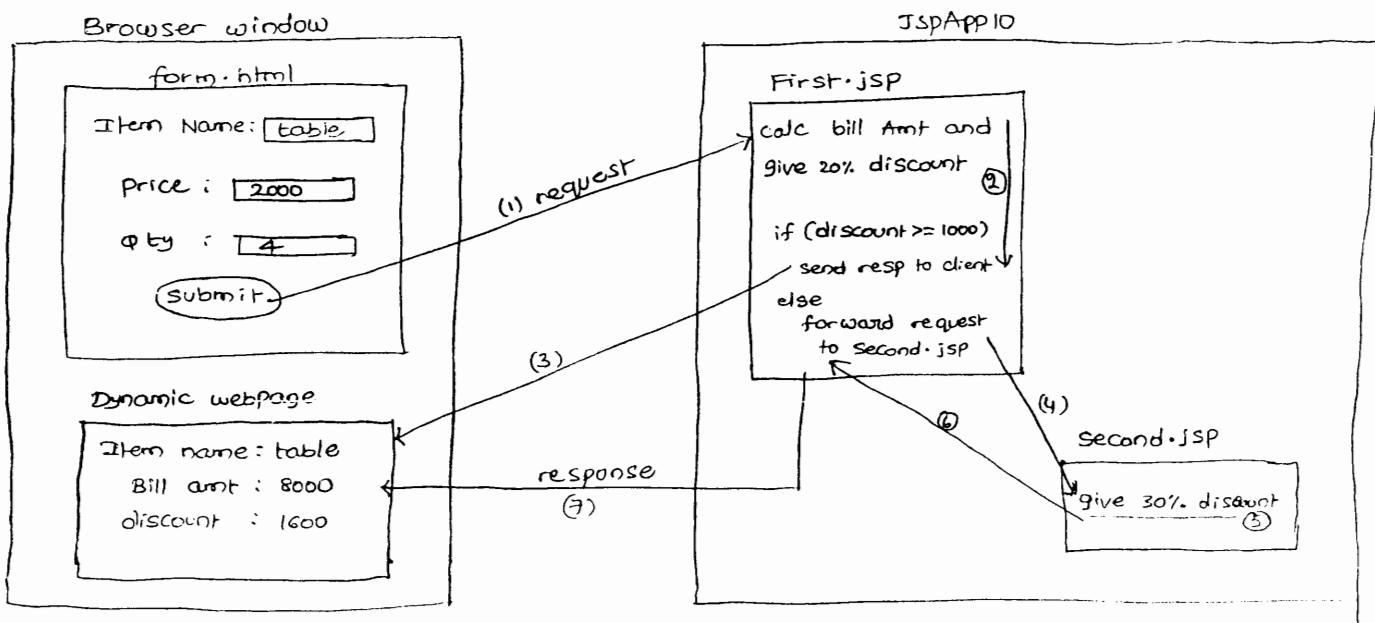
#### In B.jsp of JspApp9

<br><i> from B.jsp </i>

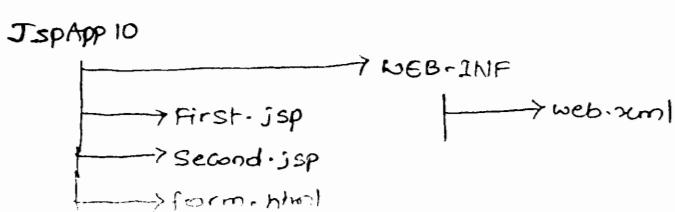
<br> Additional request parameters given by A.jsp are

<% = request.getParameter("bookname") %> } logic to read additional request  
<% = request.getParameter("price") %> parameter values

### Example application on <jsp:forward>, <jsp:param> tags



### Deployment directory structure



Procedure to develop above application by using MyEclipse IDE

NOTE: It is always recommended to execute the <jsp:forward> tag in source JSP program by enabling the Condition...

Step(1): Launch MyEclipse IDE by choosing workspace folder.

Step(2): Create web project in MyEclipse IDE.

File menu → new → web Project → JspApp10 → Finish

Step(3): Add form.html to webroot folder of the project.

Right click on webroot folder → new → JSP HTML → file name: form.html  
→ finish

form.html

```
<form action="First.jsp" method="get">  
    Item name: <input type="text" name="tname"/> <br>  
    Item Price: <input type="text" name="tprice"/> <br>  
    Item Qty: <input type="text" name="tqty"/> <br>  
    <input type="submit" value="submit"/>  
</form>
```

Step(4): Add First.jsp to the webroot folder of the project.

Right click on webroot folder → new → JSP → file name: First.jsp → Finish

First.jsp

```
<% //read form data  
    String fname = request.getParameter("tname");  
    int price = Integer.parseInt("tprice");  
    int qty = Integer.parseInt(request.getParameter("tqty"));  
    // calculate bill amount  
    int bamt = price * qty;  
    // give 20% discount.  
    float discount = bamt * 0.2f;  
    if (discount >= 1000)  
    {  
        out.println ("Item name:" + fname);  
        out.println ("Bill Amt :" + bamt);  
        out.println ("Discount :" + discount);  
    }  
    else  
    {  
        //forward the request to second.jsp %>
```

```

<jsp:forward page="second.jsp">
  <jsp:param name="p1" value="<% = fname %>" />
  <jsp:param name="p2" value="<% = bamt %>" />
</jsp:forward>
<% %>

```

} sending fname,  
bamt details  
to second.jsp from  
first.jsp as additional  
request parameter values.

Step(5): Add second.jsp to webroot folder of project.

### Second.jsp

```

<b> from second.jsp </b><br>
<% // read additional request parameters sent by First.jsp

String name = request.getParameter("p1");
int bamt = Integer.parseInt(request.getParameter("p2"));

// calculate 30% discount amount
float discount = bamt * 0.3f;

// display item details
out.println("Item name:<br>" + name);
out.println("Bill amt:<br>" + bamt);
out.println("Discount:<br>" + discount); %>

```

Step(6): Configure Tomcat Server with MyEclipse IDE

window menu → preferences → MyEclipse → servers → Tomcat → Tomcat 6.x →  
 Enable → <Tomcat-home> directory → apply → ok

Step(7): Start Tomcat server

Goto servers icon in the tool bar → Tomcat 6.x → start

Step(8): Deploy the project in Tomcat server.

Go to deploy icon in the tool bar → project → JspApp10 → add →  
 server : Tomcat 6.x → Finish → ok

Step(9): Test the application

Open browser window & type this URL

http://localhost:2020/JspApp10/form.html

There are directive include, action include tags. can you tell me why there

is no directive forward tag? Only Action forward tag is available

(A) The basic work of forwarding request operation is discarding the output of source JSP program.

The Directive tags perform their activity at translation phase by including the code of destination program to the code of JSP equivalent servlet generated for source JSP program.

In this process discarding the output of source JSP program is impossible. So there is no directive forward tag in JSP.

What is the difference between <jsp:forward> tag and response.sendRedirect() method?

(A)

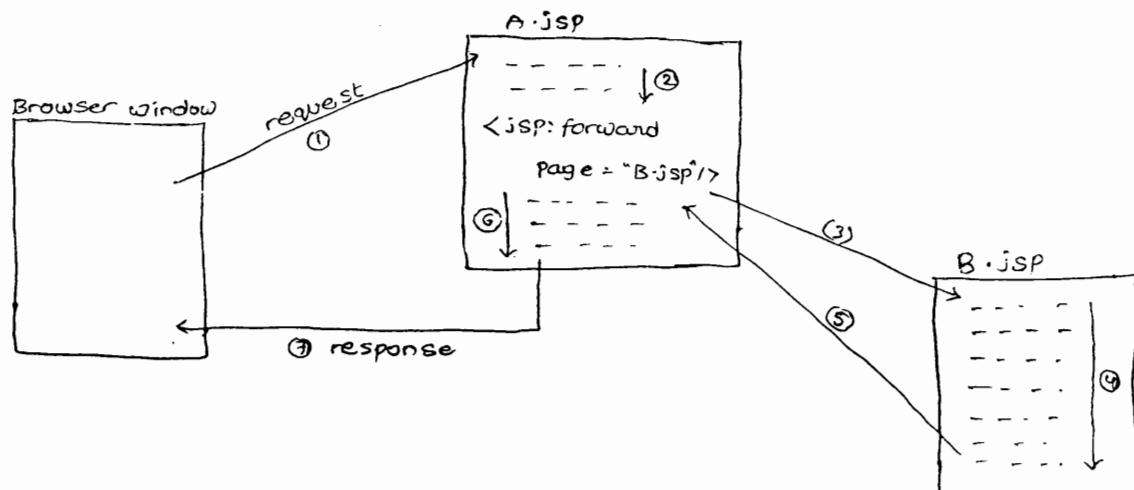


fig: Forwarding request using <jsp:forward> tag

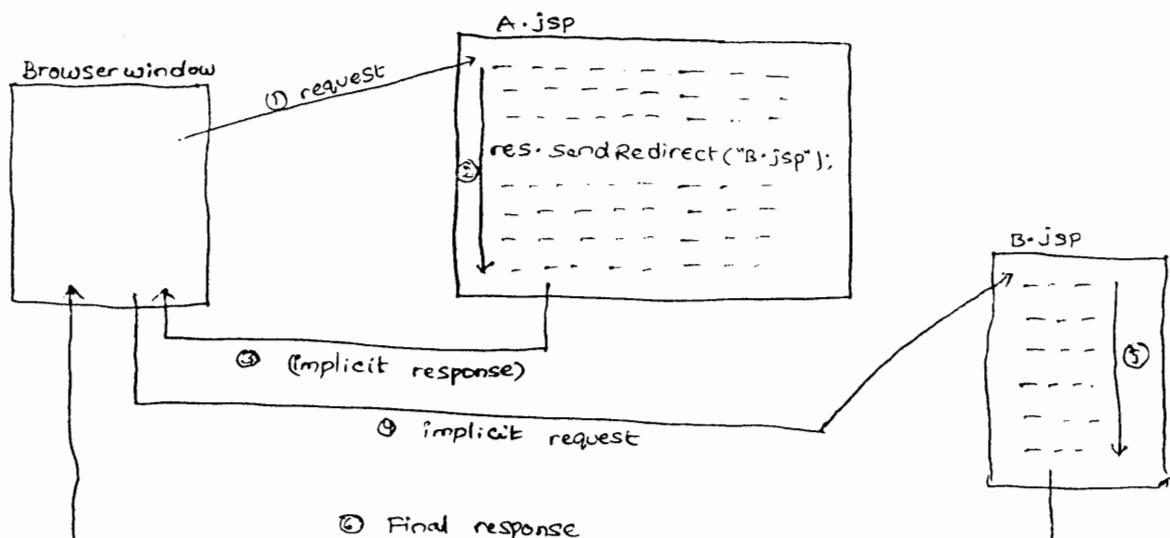


fig: send Redirection using res.sendRedirect() method.

### <jsp:forward>

- \* source JSP program communicates with destination program directly.
- \* Both source, destination programs use same request and response objects so request data coming to source JSP Program can be used in destination program.
- \* Both source and destination programs must be there in same web application.
- \* A destination program must be a html program or JSP program or Servlet program
- \* while forwarding request url in the address bar will not be changed.

### response.sendRedirect(-)

- \* Source JSP program communicates with destination program by having one network round trip with browser window.
- \* source JSP program and destination program will not use same request and response objects so the request data coming to source destination program is not visible and accessible in destination program.
- \* Both source and destination programs can be there in same web application or can be there in two different web applications of same server or different servers.
- \* Destination program can be html program or java based web resource program or non-java based web resource program.
- \* while redirecting the request the URL in the address bar will be changed into destination program related URL.

### What is the difference between the implicit objects Page & PageContext?

- (A) The implicit object Page holds "this" key word which is nothing but reference of currently invoking JSP equivalent servlet class object.

- pageContext object holds multiple details of JSP page like this, request, response objects, errorpage name, bufferSize, autoFlush mode, session objects availability status.
- JSP equivalent servlet creates / gets Access to application, config, session and out implicit objects through this PageContext object.

### Code in JSP equivalent servlet program

for implicit object page

```
Object page = this;
```

for implicit object PageContext

```
PageContext = JspFactory.getPageContext(this, request, response,
```

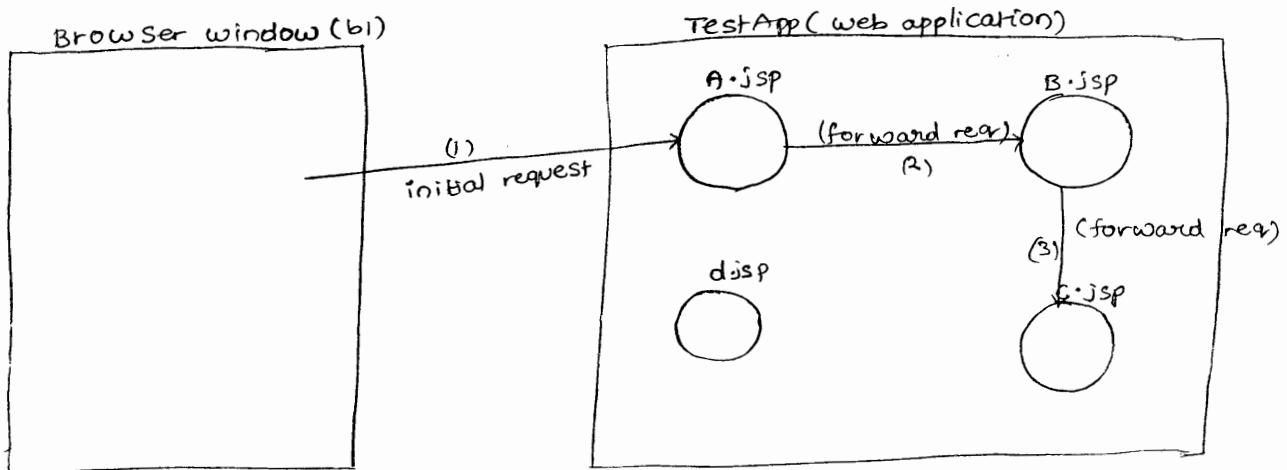
```
"abc.jsp", true, 10240, true);
```

↓                    ↓                    ↓  
 error page      Buffer size      auto flush mode  
 availability of session object.

### How to pass data between the JSP programs of web application?

- (A) (1) If source and destination programs are using same req, res objects
  - (a) use request attributes
  - (b) use additional request parameters (<jsp:param>)
- (2) If source and destination JSP programs are getting request from same browser window and not using same req and res objects
  - use session attributes
- (3) If source and destination programs are not getting request from same browser window and not using same request and response objects
  - use application attributes
- (4) use pageContext attributes

We can create pageContext attribute having pageScope (or) request scope (or) session scope (or) application scope.



\* In the above diagram the request attribute created in A.jsp is visible and accessible in B.jsp, C.jsp programs because A.jsp, B.jsp, and C.jsp programs use same request, response objects.

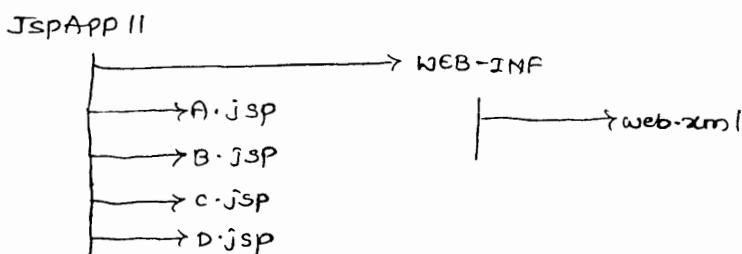
\* Request attributes are visible through out request cycle.

\* The session attribute created in A.jsp by getting request from browserwindow b1 is visible and accessible in all jsp programs of web application irrespective their request, response objects but all these web resource programs must get request from same browser window b1.

NOTE: session attributes are visible in all web resource programs of web application but they are specific to a browser window.

\* Application attribute created in A.jsp is visible and accessible in all other web resource programs of web application irrespective of the browser window from which they are getting request and irrespective of request and response objects they are using.

#### Example application



Attributes are logical names holding values the servlet Programming, JSP programming uses attributes to pass data from one web resource program to another web resource program.

setAttribute (---) → To create new attribute or to modify existing attribute values.

getAttribute (---) → To read existing attribute values.

removeAttribute (---) → To remove existing attribute

### A.jsp

```
<% //create attributes  
    request.setAttribute ("attr1", "val1");  
    request.setAttribute ("attr2", "val2");  
    application.setAttribute ("attr3", "val3");  
  
<!-- forward the request to B.jsp -->  
<jsp:forward page="B.jsp"/>
```

### B.jsp

```
<!-- read attribute values --> <b> from B.jsp </b> <br>  
attr1 (req) attribute value is = <%=request.getAttribute ("attr1") %> <br>  
attr2 (session) attribute value is = <%= session.getAttribute ("attr2") %>  
attr3 (application) attribute value is = <%= application.getAttribute ("attr3") %>  
<!-- forward the request to C.jsp -->  
<jsp:forward page="C.jsp"/>
```

### C.jsp

same as B.jsp but don't forward request anywhere

### D.jsp

same as C.jsp

### URL to test the application

give first request to A.jsp and give other request to other jsp programs either from same browser window or from different browser windows.

\* Request object can create only request scope attributes.

\* Session object can create only session scope attributes.

+ Application object can create only application scope attributes. Where as pageContext object can create page scope or request scope or session scope or application scope attributes.

### TO create pageContext attribute

PageContext.setAttribute ("attr1", "val1"); → create pageContext attribute having page scope

PageContext.setAttribute ("attr2", "val2", pageContext.SESSION\_SCOPE);

→ create pageContext attribute having page/session scope.

## other possible scopes

pageContext.PAGE\_SCOPE  
pageContext.SESSION\_SCOPE  
pageContext.APPLICATION\_SCOPE  
pageContext.REQUEST\_SCOPE

All these scopes constants of javax.servlet.jsp.PageContext class.

## To modify pageContext attribute values

PageContext.setAttribute("attr1", "val1");

→ modifies attr1 value of page scope.

pageContext.setAttribute("attr2", "val2", PageContext.SESSION\_SCOPE);

→ modifies attr2 attribute value of session scope.

## To read pageContext attribute values

String s1 = (String) pageContext.getAttribute("attr1");

→ Reads attr1 attribute value from page scope.

String s2 = (String) pageContext.getAttribute("attr2", PageContext.SESSION\_SCOPE);

→ Reads attr2 attribute value from session scope.

## To remove pageContext attributes

pageContext.removeAttribute("attr1");

→ removes attr1 attribute from page

pageContext.removeAttribute("attr2", PageContext.SESSION\_SCOPE);

→ removes attr2 attribute from session scope.

## To find attribute

String s1 = (String) pageContext.findAttribute("attr2");

→ searches and reads attr2 attribute value in multiple scopes in the following order...

a) page scope b) request scope c) application scope d) session scope.

## What is the difference between getAttribute() and findAttribute() invoked on the pageContext object?

- getAttribute() searches for the given attribute only in the specified scope (no scope is specified Page scope) whereas findAttribute() searches for the given attribute if multiple scopes in a particular order like page scope, request scope, session scope, application scope.

Revised JspApp11 application with the support of pageContext attribute

### A.jsp

```
<% // create attributes  
    pageContext.setAttribute("attr1", "val1", PageContext.REQUEST_SCOPE);  
    pageContext.setAttribute("attr2", "val2", PageContext.SESSION_SCOPE);  
    pageContext.setAttribute("attr3", "val3", PageContext.APPLICATION_SCOPE);  
    <!-- forward the request to B.jsp -->  
    <jsp:forward page="B.jsp" />
```

### B.jsp

```
<!-- //read attribute values -->  
<b> from B.jsp </b>  
attr1 (req) attribute value is = <% =pageContext.getAttribute("attr1", PageContext.REQUEST_SCOPE) %><br>  
attr2 (session) attribute value is = <% =pageContext.getAttribute("attr2") %><br>  
attr3 (application) attribute value is = <% =pageContext.getAttribute("attr3") %><br>  
<!-- forward the request to C.jsp-->  
<jsp:forward page="C.jsp" />
```

### C.jsp

Same as B.jsp but don't forward request anywhere.

### D.jsp

Same as C.jsp

\* <jsp:plugin> tag is given to display applets on the browser window.

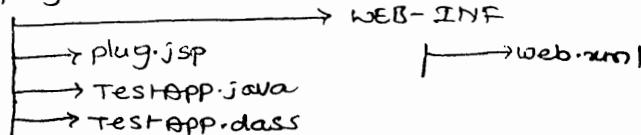
This tag internally uses <applet> tag or <object> tag of html to display applets.

\* <jsp:fallback> tag should be used as sub tag of <jsp:plugin> tag. This tag executes only when underlying browser window does not support applets.

\* The Tomcat web server internally uses <embed>, <noembed> tags for <jsp:plugin>, <jsp:fallback> tags.

### Example application

#### jsp- plug



Request URL to test the application

http://localhost:2020/jsp-plug/plug.jsp

TestApp.java

```
import java.applet.Applet;  
import java.awt.*;  
public class TestApp extends Applet  
{  
    public void paint (Graphics g)  
    {  
        setBackground(color.red);  
        g.drawString ("this is from applet to test", 50, 50);  
    }  
}
```

plug.jsp

```
<html>  
<body bgcolor="pink">  
    <jsp:plugin type="applet" code="TestApp.class" codebase="/jsp-plug" width=300  
                (*!) height=300>  
        <jsp:fallback> browser does not support applets </jsp:fallback>  
    </jsp:plugin>  
</body>  
</html>
```

web.xml

```
<web-app>
```

(\*) → specifies the directory where given applet class is available.  
(TestApp.class)

Java Bean :

- \* A java bean is a java class that contains getter and setter methods. There is no special key word to develop the java class as java bean.
- \* The member variables of java bean class are technically called as bean properties and every bean property contains one getter and one setter method.
- \* Getter methods are useful to get or read the data from bean properties. whereas as Setter methods are useful to write the data to bean properties.

Ex:

```
public class StudentBean
```

```

    //bean properties
    private int sno;
    private String sname;
    private float avg;

    // write getXXX() and setXXX(-) methods
    public void setSno (int no)
    {
        this.sno = no;
    }
    public int getNo()
    {
        this.sno;
    }
    public void setSname (String sname)
    {
        this.sname = sname;
    }
    public String getSname()
    {
        return sname;
    }
    public void setAvg (float avg)
    {
        this.avg = avg;
    }
    public float getAvg()
    {
        return avg;
    }
} //class

```

\* If java programmers follow java bean standards while developing their java classes then it becomes easy for the programmers to exchange the java classes.

\* Java bean is a standard to develop java classes having getXXX() and setXXX(-) methods.

\* The best way to develop re-usable simple java class is developing that class as java bean class.

What is the difference between Java bean and EJB (Enterprise Java Bean)?

### Java bean

\* An ordinary java class containing getter and setter methods.

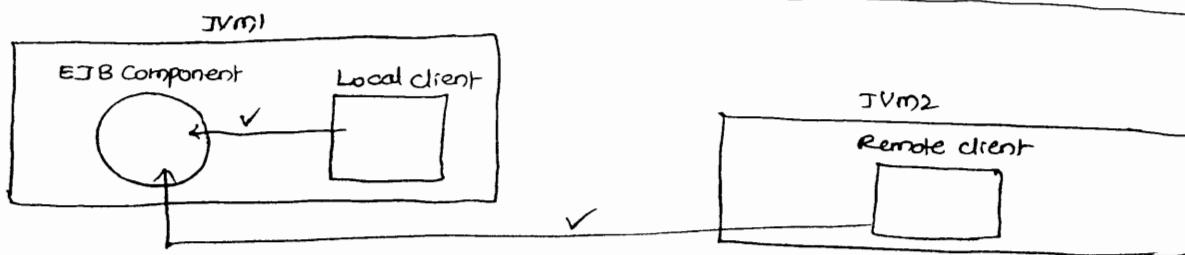
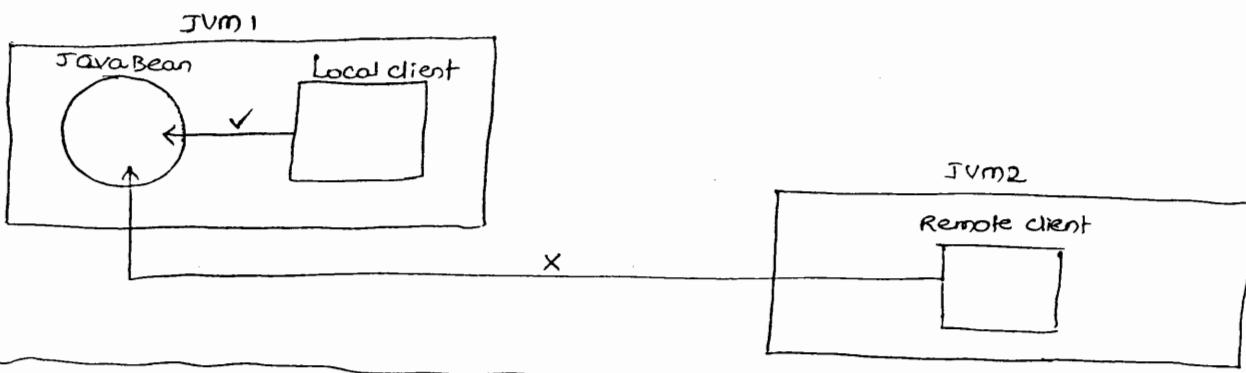
\* Just needs JVM for execution

### EJB

\* Distributed technology develop distributed applications.

+ Needs JVM and EJB container for execution.

- \* No life cycle methods.
- \* Life cycle methods are there.
- \* Useful as a reusable helper class in projects.
- \* Useful as technology to develop business components of the project.
- \* Allows only local clients.
- \* Allows both local and remote clients.
- \* In one computer multiple JVMs can be there to run multiple Java applications simultaneously or parallelly. If application and its client resides in the same JVM then that client is called local client to application. If application and its client resides on two different JVMs of same computer or different computers then that client is called remote client to application.



- \* A Java Bean class can have 3 types of Bean properties.
  - Simple Bean properties
  - boolean Bean properties
  - Indexed Bean properties

#### Simple Bean properties

- \* Allows to store only one value at a time.

Ex:

```

int sno;
public void setSno(int sno)
{
    this.sno=sno;
}
public get getSno()
{
    return sno;
}
  
```

### Boolean Bean properties

\* Allows to store only true or false value.

Ex: boolean married;

```

public void setMarried(boolean married)
{
    this.married = married;
}

public boolean getMarried() / isMarried()
{
    return married;
}

```

### Indexed Bean properties

\* Allows to store multiple values . This bean property type is an array.

Ex: String[] colors;

```

public void setColors(String[] colors)
{
    this.colors = colors;
}

public String[] getColors()
{
    return colors;
}

```

\* For servlet program to javaBean communication the servlet program should be an object of javabean class and should call methods on that object.

\* For jsp program to java bean communication we can use the following tags.

<jsp:useBean>  
<jsp:setProperty>  
<jsp:getProperty>

### java based web application development models

#### 1. Model 1 architecture

→ use only servlets or only JSP programs as server side web resource programs of web application

#### 2. Model 2 architecture

(MVC Architecture)

##### 1. MVC1 architecture

→ In MVC1 and MVC2 architectures multiple technologies support will be taken to develop web applications.

## Logics that can be there in our java web applications

- (1) Request Data gathering logic
- (2) form validation logic
- (3) Business logic
- (4) session management logic
- (5) persistance logic
- (6) middleware services logic
- (7) presentation logic and etc...

\* Request Data gathering logic is responsible to gather various details given by client in its httprequest like form data, header values and etc. For this, use `getXXX()` methods on request object.

\* In form validation logic checks the format and pattern of the form data.

Ex: checking whether email id is having @, . symbols or not.

\* Business logic processes the request and generates the results.

\* Business logic is main logic of web application.

\* Session management logic remembers the client data across the multiple requests during a session through session tracking techniques like hidden form fields, cookies, HttpSession with Cookies and HttpSession with URL rewriting and etc...

NOTE: There are no special tracking techniques in JSP. JSP should also use the above said four session tracking techniques.

\* persistance logic is responsible to manipulate database data by performing "curd" operations. (Ex: JDBC code)

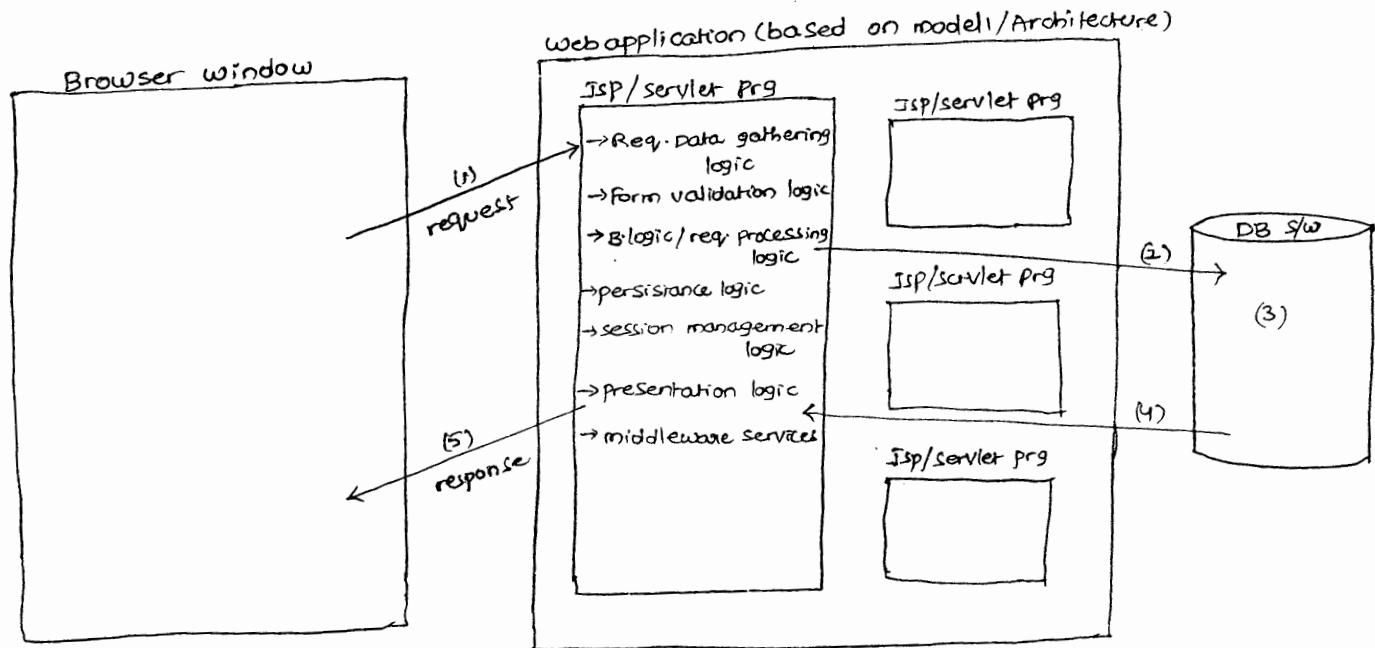
\* middleware services are the additional services which can be applied on the applications like JDBC Connection pooling, security and etc...

\* presentation logic generates user interface required for end user. It is also responsible to format the results and to display the results. (html code).

## Model - I architecture

\* In this model either servlet programs or jsp programs will be used as server side web resource programs. more ever if servlet programs are used then JSP programs will not be used and viceversa.

\* In every server side program multiple logics will be mixed up.



\* In model1 architecture web application, the client side web resource programs can be there but either servlets or jsp must be used as server side programs.

\* Most of the programmers prefer working with JSPs + while developing model1 architecture base web application.

#### \* Disadvantages of model1 architecture

+ multiple logics will be mixed up in every server side web resource program. This indicates there is no clean separation of logics and this also indicates modification done in one logic may disturb other logic.

+ parallel development is not possible so the productivity is very poor.

NOTE: doing more work in less time with good accuracy gives good productivity.

+ middleware services (some) must be implemented by the programmer manually. This improves burden on the programmer.

#### Advantages

+ knowledge on either servlet or jsp is sufficient to develop web resource programs

\* since parallel development is not possible multiple programmers are not required to develop the web applications.

---

+ to overcome the drawbacks of model-1 architecture use model-2 (or) MVC architecture to develop the web applications.

## MVC / model-2 architecture

M → model Layer → Represents b. logic + persistance logic

→ use java class/ java bean/ ejb components/ spring App /  
Spring with HB App in model layer to develop logics

→ It is like Accounts officer.

V → view Layer → Represents presentation logic

→ use html/ jsp programs to develop this presentation logic.  
→ It is like Beautician.

C → controller Layer → Represents Integration logic/ connectivity logic

→ use servlet program or servlet filter program to develop these  
logics.

→ It is like Traffic police/ supervisor

\* Integration logic is responsible to take the request from browser window,  
to pass that request to model layer resource, to gather result from model layer  
resource and to pass the result to view layer resource.

\* Integration logic is responsible to control and monitor every operation of the  
web applications execution.

\* In model 2/mvc architecture based web application development multiple technologies  
support will be taken by programmers.

\* In model MVC1 architecture the same servlet programs/ JSP programs will act  
as view layer, controller layer resources representing presentation, business logics  
and separate resources will be taken for model layer.

\* In mvc2 architecture three different resources will be taken in 3 different  
layers.

JSP programs → In view layer

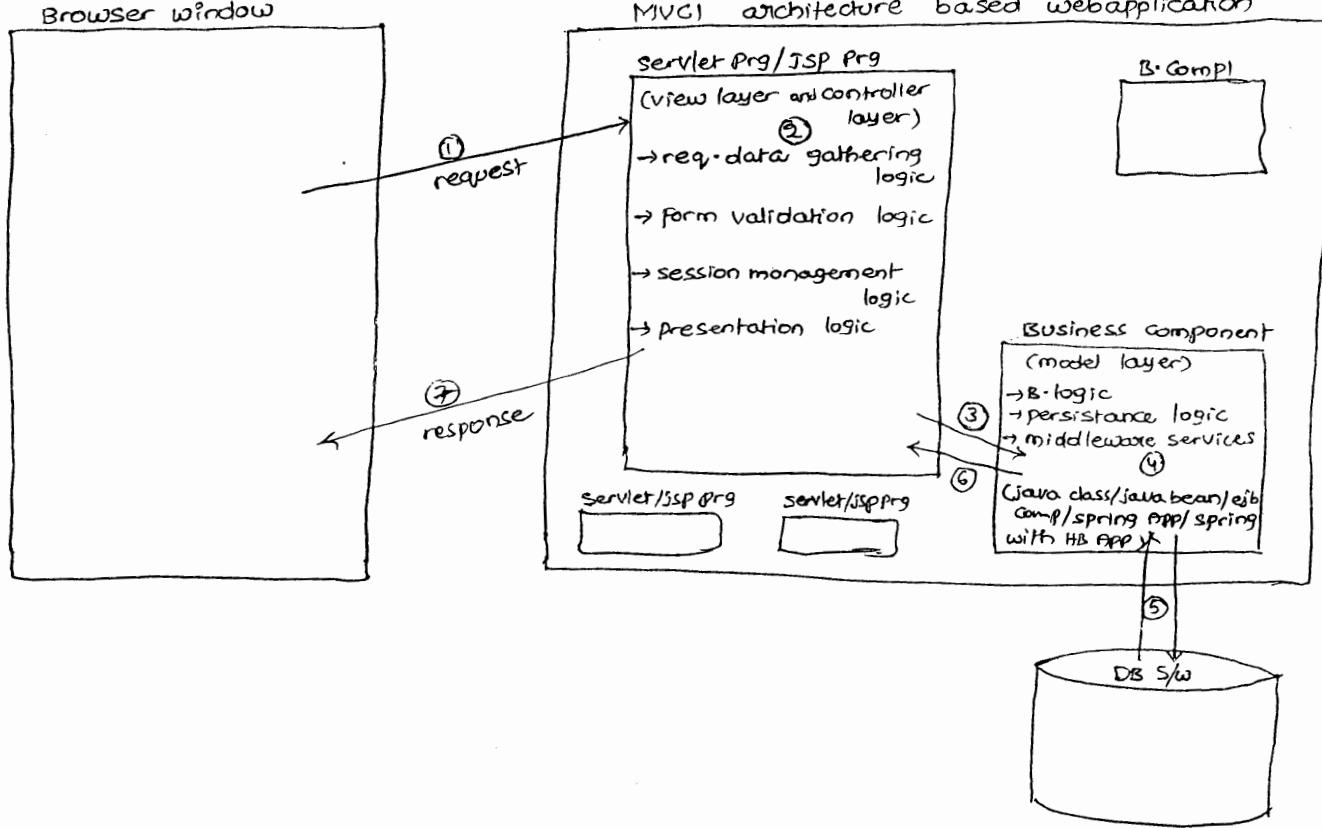
servlet program → In controller layer

java class/ javabean/ ejb Component/... → In model layer

\* In mvc1 architecture multiple resources can be there in each layer.

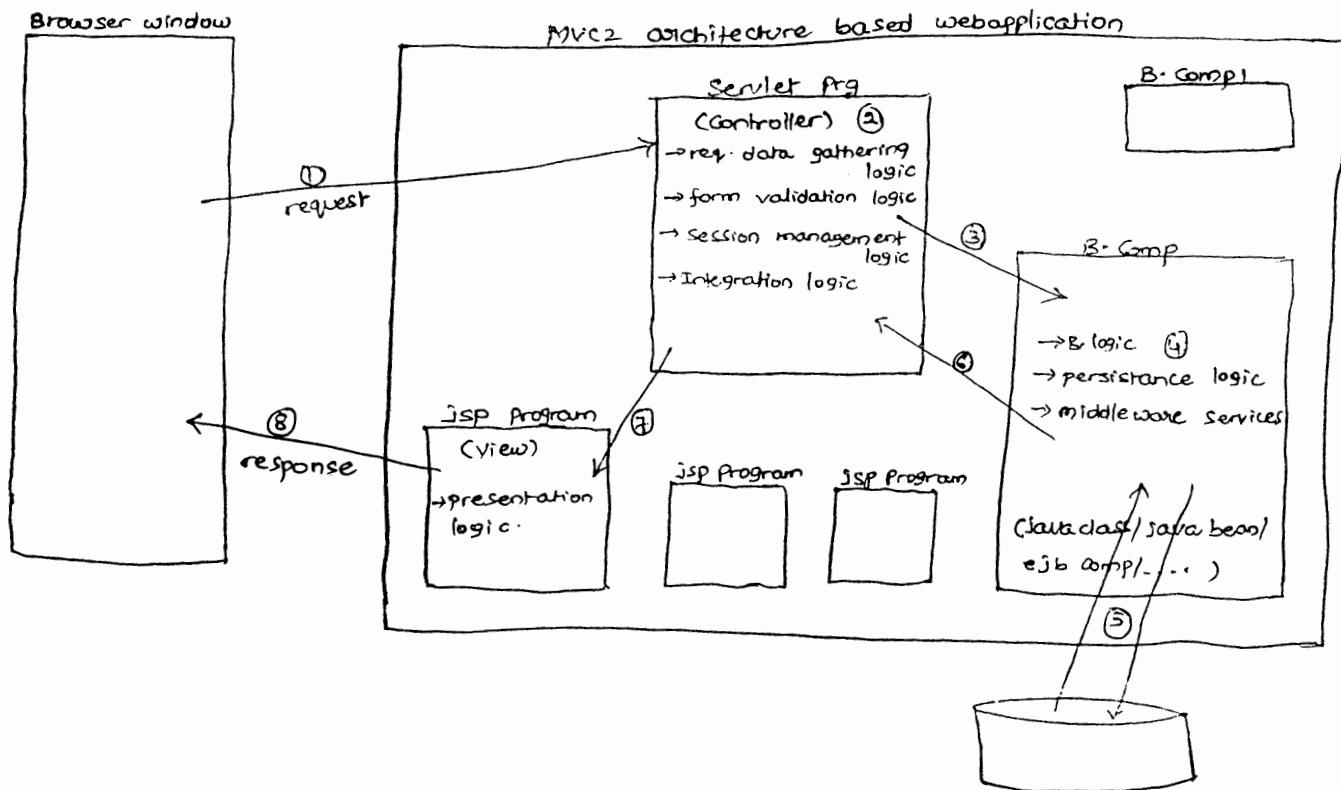
\* In mvc2 architecture multiple resources can be there in view, model layers but  
only one servlet program will be in controller layer.

## MVC-1 architecture



\* MVC1 gives quite better clean separation of logics compare to model-1 because it is separating business logic and persistence logic from servlet program/JSP program. But for more clean separation of logics it is recommended to use MVC2 architecture.

## MVC-2 architecture



with respect to the diagram

- ① Browser window gives request to web application.
  - ② As a controller servlet program traps and takes the request
  - ③ Servlet program passes the request to model layer business component by using integration logic.
  - ④, ⑤ → The business logic of business component process the request and generates the results. Also talk talks with DB if necessary.
  - ⑥ → Business Component generated results goes back to Servlet program (Controller)
  - ⑦ The controller Servlet program passes the results to the view layer Jsp program.
  - ⑧ The JSP program uses the presentation logic to format the results and sends formatted result to browser window as response.
- + use model-1 architecture to develop small scale web application.  
+ use MVC-1 architecture to develop medium scale web application.  
+ use MVC-2 architecture to develop large scale web application.

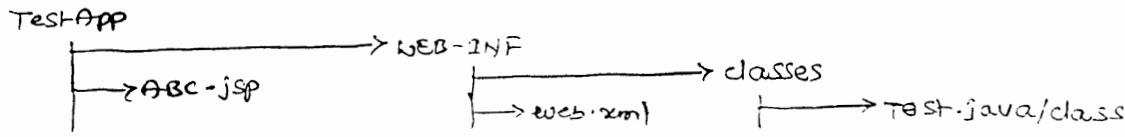
### Advantages of MVC-2 architecture

- (1) There are multiple layers in web application development so we can achieve clean separation of logics.
  - (2) Modifications done in logics of one layer does not effect logics of another layer.
  - (3) Parallel development is possible so productivity will be good.
  - (4) Provides easiness in enhancement and maintenance of the web application.
  - (5) MVC-2 is a industry standard to develop the web applications.
  - (6) When spring, hibernate, EJB technologies are used in the model layer we can get the benefit from middleware services (built-in). This reduces burden on the programmer.
- ⇒ MVC-2 is industry standard to develop the web applications.

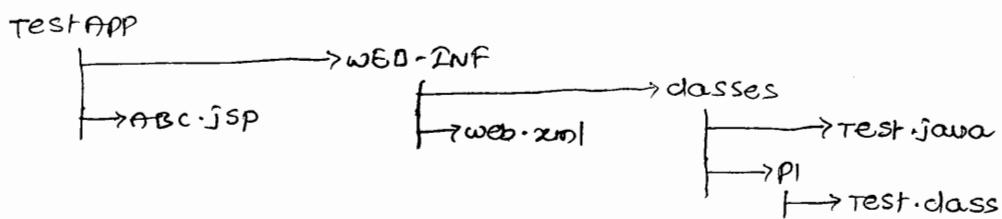
### disadvantages

- + For parallel development multiple programmers are required.  
+ Knowledge in multiple technologies is required to develop the web applications.

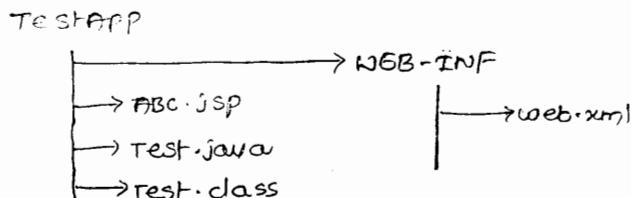
- while we are developing MVC2 architecture based web applications along with taking support from multiple technologies we should also implement set of rules which are called MVC-2 rules or principles.
- (1) Every layer is given to have certain logics. Just place only those logics and don't place any excess logics.
- (2) It is recommended to use JSPs in view layer, servlets in controller layer. Use javaBean or java class in model layer when business logic is less.
- (3) Use EJB or spring with HB in model layer when business logic is huge.
- (4) Every operation in the web application must take place under the controller of controller servlet.
- (5) There can be multiple resources in the view layer and multiple resources in model layer but there must be only one controller servlet or controller ServletFilter program in controller layer.
- (6) The view layer resources must not talk with model layer resources directly and vice versa. That means they must communicate with each other through controller servlet.
- (7) Two resources of view layer (JSP programs) must not talk with each other directly. That means they must communicate with each other through controller layer.
- \* In a web application where JSP to Java bean communication is taking place with the support of <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty> tags comes under MVC-1 architecture based web application development.
- \* A Java bean class can also contain business methods along with getter and setter methods.
- \* If you want to use Java class or Java bean in JSP program then it must be placed in a package defined in WEB-INF\classes folder. otherwise it must be placed along with JSP program. If that Java class is directly there in WEB-INF\classes folder then it can not be used in JSP program.



Test class can not be used in ABC.jsp



Test class can't be used in ABC.jsp



Test class can be used in AB.jsp

### <jsp:useBean>

Syntax . <jsp:useBean attributes/>

used for creating/locating an object of javabean class

#### attributes

id = " " //instance name (object name)

class = " " // a fully qualified javabean class name

scope = "page/session/request/application" (default scope is "page")  
//specifies the scope of the bean class obj

#### page (default)

bean class obj is available throughout the request page.

#### session

bean class obj is available throughout the request session

#### request

throughout the request

#### application

available for all pages within the context (web application)

beanName = "logical name of java bean"

type = "reference class type of java bean class object".

Ex: <jsp:useBean id="st" class="pi.StudentBean" scope="session" />

Creates pi.StudentBean class obj having name "st" and keeps in session scope

If this object is already available in session scope then it locates that object from session scope. For this it internally uses `pageContext.setAttribute()` and `pageContext.getAttribute()` methods.

The above statement creates object like this:

`PI.StudentBean st = new PI.StudentBean();`

"st" object type is `PI.StudentBean`

"st" <sup>reference</sup> object type is `PI.Student Bean`

Ex:

`<jsp:useBean id="st" type="ABC" class="PI.StudentBean" scope="session" >`

creates or locates `PI.StudentBean` class object (`st`) <sup>from request scope</sup>.

Bean class object creation statement

`ABC st = new PI.StudentBean();`

"st" → reference type is ABC class

"st" → object type is `PI.StudentBean` class

→ Here `PI.StudentBean` class extends from ABC class.

\* `<jsp:useBean>`, `<jsp:setProperty>`, `<jsp:getProperty>` are 3 independent tags.

`<jsp:setProperty>`

calls `setXXX` methods on java class object to set given values to bean properties.

Syntax: `<jsp:setProperty attribute(s) />`

Attributes

`property = " "` // bean property name

`name = " "` // (bean class obj name) id attribute value given in the `<jsp:useBean>`

`value = " "` // value to be set for bean property

`param = " "` // `input(request)` parameter name

NOTE: value or param → any one attribute has to be used at a time.

Ex: `<jsp:setProperty name="st" property="sno" value="567" />`

→ This internally calls `setSno()` method on `st` obj and assigns value "567" to the bean property `sno`.

Ex: <jsp:setProperty name="st" property="sname" param="sname"/>  
this internally calls setSname() method on st obj to assign the value  
of "sname" reg param to sname bean property.

### <jsp:getProperty>

calls getXxx() methods to read values from bean properties.

syntax: <jsp:getProperty attributes/>

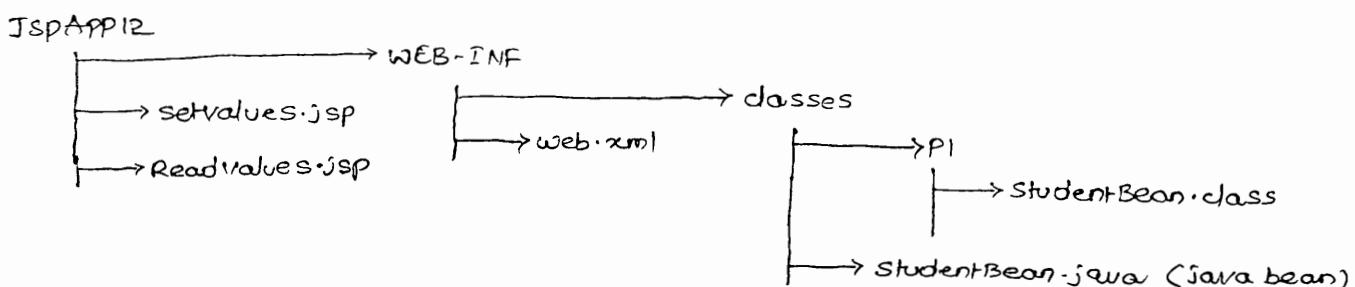
#### attributes

name = " " // (bean obj name) id attr value given in <jsp:useBean>  
Property = "bean property name"

Ex: <jsp:getProperty name="st" Property="sno"/>

gives "sno" bean property value by calling getSno() method on bean class  
object ("st") and also displays that value on browser window.

### Example application on Jsp to Java bean communication (MVC-1 architecture)



### StudentBean.java

```
package PI;
public class StudentBean
{
    // bean properties
    private int sno;
    private String sname;
    private float avg;
    // Constructor
    public StudentBean()
    {
        System.out.println("StudentBean o-param constructor");
    }
    // write getXxx() and setXxx() methods
```

```

public void setSno(int sno)
{
    System.out.println("Student Bean: setSno (-) method");
    this.sno = sno;
}

```

=====

3

> javac -d . StudentBean.java

### SetValues.jsp

```

<%@ page import = "P1.StudentBean" %>

<!-- create or locates java bean class obj -->
<jsp:useBean id = "st" class = "P1.StudentBean" scope = "session" />

<!-- Set values to bean properties -->
<jsp:setProperty name = "st" property = "sno" value = "101" />
<jsp:setProperty name = "st" property = "sname" value = "rajesh" />
<jsp:setProperty name = "st" property = "avg" value = "67.89" /> <br> values are
set to bean properties

```

### ReadValues.jsp

```

<%@ page import = "P1.StudentBean" %>

<!-- create or locate bean class obj -->
<jsp:useBean id = "st" class = "P1.StudentBean" scope = "session" />

<!-- read and display bean properties values on browser window -->
sno value is: <jsp:getProperty name = "st" property = "sno" /> <br>
sname value is: <jsp:getProperty name = "st" property = "sname" /> <br>
avg value is: <jsp:getProperty name = "st" property = "avg" />

```

### web.xml

<web-app>

\* Irrespective of whether thread safety is enabled or not enabled (isThreadSafe = "true" / "false") the jsp equivalent servlet manages session scoped, application scoped javabean class objects having thread safety with the support of synchronized blocks.

URL to test the application:

<http://localhost:2020/JspApp12/SetValue.jsp>

<http://localhost:2020/ReadValues.jsp>

\* we can use the param attribute of <jsp:setProperty> tag to set the received request parameter value as bean property value.

```
<jsp:setProperty name="st" property="sno" param="sno"/>
```

```
<jsp:setProperty name="st" property="sname" param="sname"/>
```

```
<jsp:setProperty name="st" property="avg" param="stavg"/>
```

<br> values are set to bean properties ↓  
bean property name

Request URL is

http://localhost:2020/JspApp12/SetValues.jsp ? sno=3456&sname=ramesh&stavg=37.5  
query string representing req.params

\* If request parameter names are matching with bean property names then we can use property = "\*" in <jsp:setProperty> tag to set request parameter values as bean property values.

#### SetValues.jsp

```
<%@ page import="pi.StudentBean" %>
```

```
<jsp:useBean id="st" class="pi.StudentBean" scope="application"/>
```

```
<jsp:setProperty name="st" property="*"/>
```

<br> value are set to bean properties.

\* The application that can display and rotate advertisements / Ads in web pages is called as AdRotator.

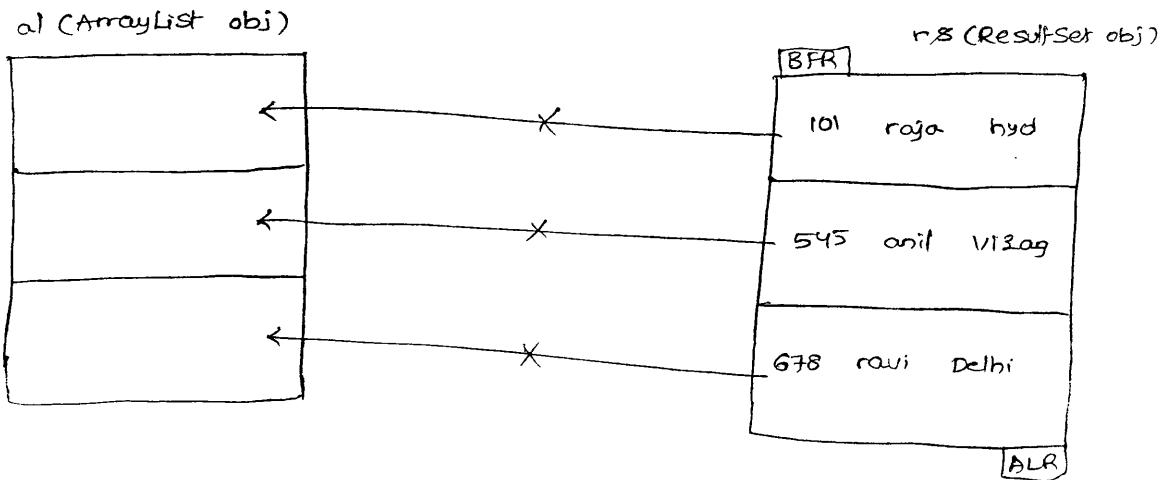
\* Our AdRotator application is having the following features.

- (1) Should Render the advertisements as graphical hyperlinks pointing to client's related web sites.
- (2) The advertisements must be changed automatically at regular intervals.
- (3) Advertisement should be rendered randomly in the web page.
- (4) Develop application based on MVC architecture.

\* For example application on advertisement rotator refer application - 6 of page nos 110 and 111.

## Mini Projects Discussion

- \* In order to send Java objects over the network it must be developed as serializable object.
- \* Java object becomes serializable object only when its class implements java.io.Serializable interface directly or indirectly.
- \* All Collection framework data structure objects are serializable objects by default.
- \* ResultSet object is not serializable object so we can not send that object over the network. But in multilayer environment like MVC we need to send the ResultSet object data over the network.  
above said
- \* To solve the \* ResultSet problem there are two solutions.
  - solution(1): use RowSets as alternate for ResultSets
  - solution(2): copy the data of ResultSet obj to Collection Framework data structures and send them over the network.
- \* RowSets are serializable objects so we can send them over the network but very few JDBC drivers support for RowSets. Due to this industry prefers working with solution(2) to send ResultSet object data over the network.
- \* If you are performing only read operations on data structures after receiving them then it is recommended to work with non synchronized data structures like ArrayList, HashMap for better performance.
- \* If you want to perform read write and other operations on the received Collection Framework data structure then it is recommended to work with synchronized data structure like Vector, Hashtable to avoid multithreading related concurrency issues.
- \* We can not move ResultSet object records directly to ArrayList object elements on each record to each element basis because in one element of ArrayList object we can store only one Java object at a time. But in each record of ResultSet there is a possibility of having multiple objects and other values.



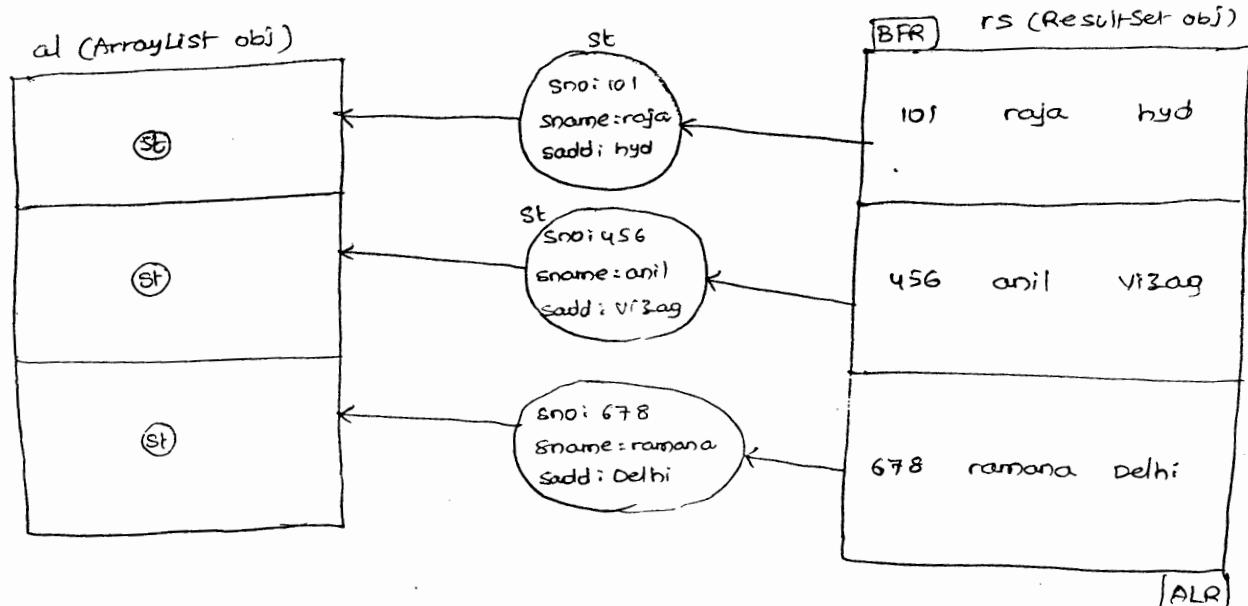
\* TO solve the above problem store each record values of ResultSet in a user defined Java class object and add that object to each element of ArrayList object. Here this user defined java class is a Java Bean and it is technically called as DTO class or VO class (Value Object).

#### D.T.O class or V.O class

```
public class StudentBean implements java.io.Serializable
{
    private int Sno;
    private String Sname;
    private String Sadd;
    //write setXXX(-) and getXXX() methods
    ===
}
```

Logic to transfer ResultSet object records to ArrayList object elements D.T.O class Support

```
ResultSet rs=st.executeQuery("select * from student");
ArrayList al = new ArrayList();
while(rs.next())
{
    Student Bean st=new Student Bean();
    /add each record values to D.T.O class obj
    st.setSno(rs.getInt(1));
    st.setSname(rs.getString(2));
    st.setSadd(rs.getString(3));
    /add D.T.O class obj to ArrayList
    al.add(st);
}
```



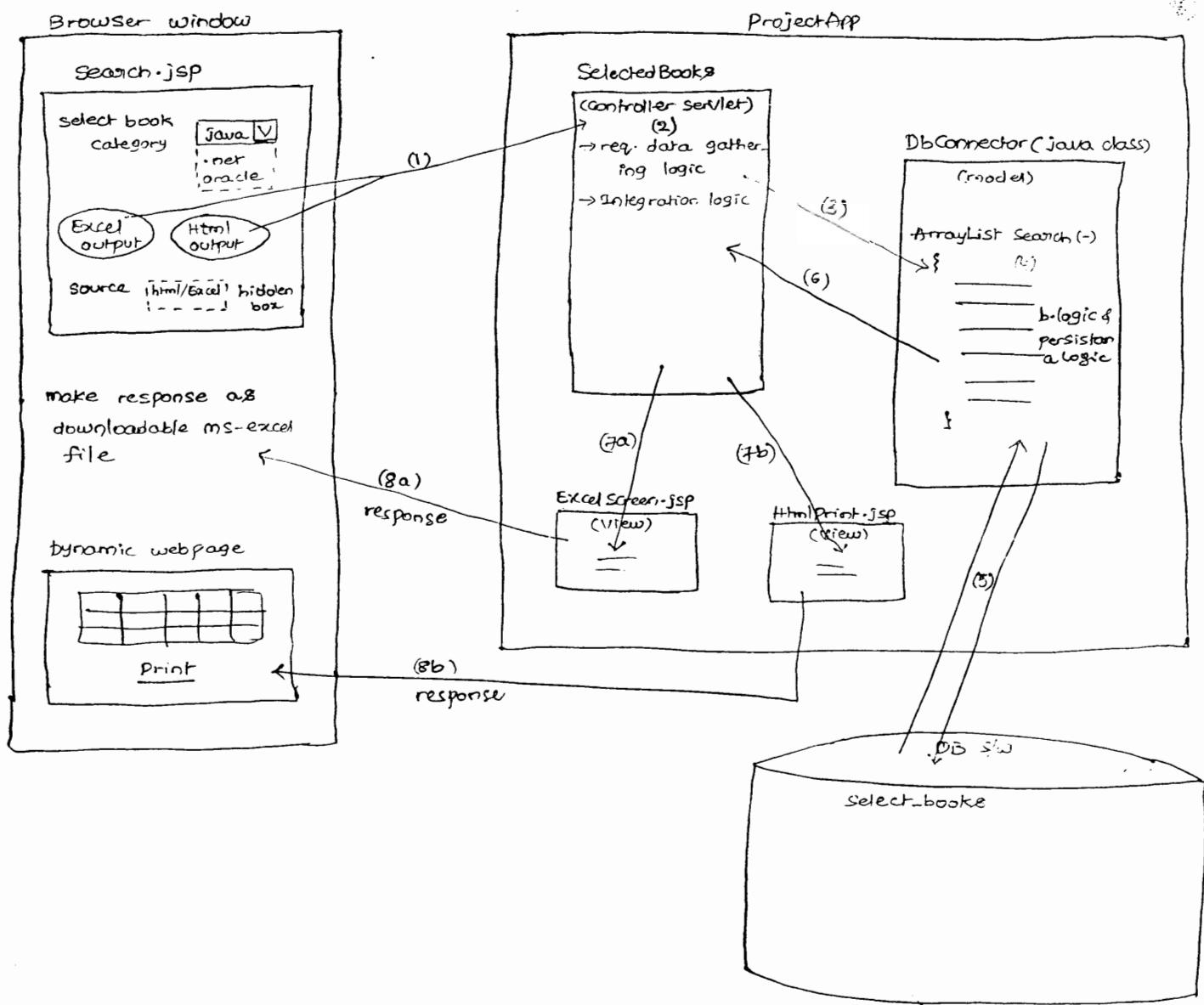
where did you use java bean in your project ?

- (a) (i) As model layer resource in MVC-1 , MVC-2 architecture based projects.
  - (ii) As D.T.O class or V.O class while transferring ResultSet object to Collection framework datastructure as shown above.
  - (iii) As HB peristence class in HB Programming.
  - (iv) As spring Bean in spring programming and etc .
- 

\* can you tell me where exactly you have utilized the collection framework data structures in your project?

- (a) \* `java.util.Properties` is required to map while maintaining JDBC driver and database details outside the JDBC application.
  - \* `ArrayList` or `Vector` support will be taken while trying to send ResultSet object data over the network .
  - \* To maintain JNDI properties, mail properties of Java mail api map data structures are required .
  - \* To maintain application data ( huge amount) within the application Collection framework data structures are required .
  - \* To maintain some form data/page data across the multiple requests as a single value during the session Collection framework data structures are required .
-

NOTE: To send collection framework datastructure over the network the objects that added to datastructures as elements must be serialisable objects.



- (1) Enduser selects the book category and submits the request to web application.
- (2) The Controller Servlet traps and takes the request and reads form data from the form page.
- (3) Controller servlet uses integration logic to call search business method of DbConnector class having the selected book category as argument value.
- (4) The Search business method uses business logic and persistiance logic to gather book details from db table into ResultSet object and moves
- (5) The ResultSet object is passed to the DbConnector class.
- (6) The DbConnector class uses its business logic and persistiance logic to interact with the database.
- (7a) The DbConnector class returns the ResultSet object to the Controller servlet.
- (7b) The Controller servlet passes the ResultSet object to the ExcelScreen.jsp view.
- (8a) The ExcelScreen.jsp view generates the Excel output file.
- (8b) The ExcelScreen.jsp view generates the HTML output file.

that ResultSet object data to ArrayList by taking the support of D-T-O class.

(6) Search business method sends the ArrayList object back to the controller servlet.

(7a), (7b) The controller servlet sends ArrayList to result pages (ExcelScreen.jsp or HtmlPrint.jsp) as request attribute value.

(8a), (8b) Result pages sends response to browser window.

NOTE(1): ExcelScreen.jsp sends response to client as downloadable MS-Excel file where as HtmlPrint.jsp sends response to browser window as printable html table content.

The following observations can be done in the above project.

(i) web application is developed based on MVC2 architecture.

where

M → Model Layer → java class (DbConnector)

V → View Controller → jsp programs (3 prgs)

C → Controller Layer → Servlet Prg (SelectedBooks)

(i) Form Page is submitting request to web application through ordinary buttons taking the support of javascript.

(ii) Multiple button generated request related logics are differentiate in web application.

(iii) JavaScript also used for form validations (selecting item from in select box is mandatory).

(iv) D-T-O class support is taken to transfer ResultSet object data to ArrayList.

(v) Servlet chaining support is taken by controller servlet communicating with result pages and request attributes are used to pass data from controller servlet to result pages.

(vi) Response is made as downloadable ms-excel file and also made as the printable web page.

\* For the source code of above project refer page nos 145-150.

\* For other mini projects refer page nos 132, 133.

## Custom JSP Tag Library:

- \* JSP tag library is a library that contains set of JSP tags.
- \* Programmers can work with three types of tag libraries in their JSP programs.
  - (1) Built-in tag libraries of JSP technologies.  
(All Standard Action tags of JSP)
  - (2) Third party supplied JSP tag libraries  
Ex: Struts tag libraries, JSF tag libraries, JSTL and etc...
  - (3) custom JSP tag libraries  
(developed by programmers)
- \* According to industry standard it is recommended to develop JSP programs as Java codeless JSP programs to increase readability and easyness in JSP program. That means we should not use scriptlet, declaration, expression tags in our JSP programming.
- \* The built-in tags of JSP are not at all sufficient to develop JSP programs as Java codeless JSP programs so we need to think about developing custom tag libraries or using third party tag libraries.
- \* In custom JSP tag libraries all JSP tags are custom JSP tags.
- \* The Java class that defines the functionality of a JSP tag is called as tag handler class. For this the class must extend from javax.servlet.jsp.tagext.TagSupport class.
- \* Every JSP tag library contains one tld file (Tag Library Descriptor) having the configuration details of JSP tag like JSP tag name, attribute names, tag handler class name, list of possible values for attributes and etc.

## Procedure to develop custom JSP tag library

Step 1: Design the tag library and its JSP tags.

<ABC> → Should print 1 - 10 numbers

<XYZ> → Should print 10 - 1 numbers

Step 2: Develop tag handler classes defining the functionalities of JSP tags in WEB-INF\classes folder

ABCTag.java

```

public class ABCTag extends TagSupport
{
    public int doStartTag()
    {
        == (g)
    }

    public int doEndTag()
    {
        == (g)
    }
}

```

### XyzTag.java

```

public class XyzTag extends TagSupport
{
    public int doStartTag()
    {
        == (g)
    }

    public int doEndTag()
    {
        == (g)
    }
}

```

NOTE: The logic of doStartTag() executes when open Jsp tag encounters  $\langle \text{name} \rangle$  in this method definition we can use method definitions and body of Jsp tags to define the functionality of Jsp tag. where as the doEndTag() executes when  $\langle \text{name} \rangle$  encounters. In this method definition we can define the finishing functionalities.

\* doStartTag(), doEndTag() methods are container call back methods ~~and/or~~ life cycle methods of Tag handler classes.

Step(3): Configure all the Jsp tags of custom Jsp tag library in tld file.

### WEB-INF\ sathyam.tld (xml file) (Blue print)

$\langle \text{ABC} \rangle \xrightarrow{\text{(g)}} \text{ABCTag.class}$

$\langle \text{XYZ} \rangle \rightarrow \text{XyzTag.class}$

\* Step(1) to step(3) talks about custom Jsp tag library.

Step(4): Configure Jsp tag library in web.xml file of web application.

```

<web-app>
    <taglib>
        (d)
        <taglib-uri>demo</taglib-uri>

```

```
<taglib-location> /WEB-INF/sathyas.tld </taglib-location>  
</taglib>
```

(e) name & location of tld file

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

NOTE: Every JSP Taglibrary that is configured in web.xml file will be identified through its `<taglib-uri>` (like demo).

Step 5: Use the JSP tags of tag library in JSP programs of web application.

Test.jsp

```
<%@taglib uri="demo" prefix="st"%>  
<st:ABC/>  
<st:XYZ/>
```

! (c)      ! (c)  
↓      ↓  
Collect from web.xml      User defined  
(a)      (b)  
→ (b): output of `<st:ABC>` tag goes to browser window.

NOTE: Prefix of the tag is useful to identify the tag library of JSP tag. When two JSP tags of two different JSP tag libraries contains same name and different functionalities in order to use both JSP tags in a single JSP program we can use prefixes to differentiate those tags.

\* All custom JSP tags must be developed as action tags so we must use prefixes while working with those tags.

\* In the above code the alphabets (a) to (h) indicates flow of execution related to `<st:ABC>` tag.

- (a) → The `<st:ABC/>` tag encounters in the execution of JSP program.
- (b) → The prefix st will be gathered from the tag.
- (c) → Based on the prefix the `<taglib-uri>` demo will be gathered from `<%@taglib ...>` tag (This tag is useful to specify the `<taglib-uri>` of tag library whose tags are going to be used in current JSP program)
- (d), (e) → From web.xml file the tld file name and location will be gathered based on `<taglib-uri>`.
- (f) → From tld file the tag handler class of `<ABC>` tag will be gathered (ABC Tag).
- (g) → The code placed in tag handler class executes through container callback methods.
- (h) → The output generated by Taghandler class goes to browser window as the output of `<st:ABC>` tag.

- \* In every tag handler class the pageContext object is visible because it is protected member variable of pre defined tag support class.
- \* Programmer uses this ~~it's~~ inherited pageContext object in Tag handler class to create/ to get access through other implicit objects of JSP like out, request, response and etc..
- \* For example application on custom JSP tag library development refer application given in page nos 129 - 132.

### JSTL

- \* The built-in JSP tags of JSP technology are not sufficient to make JSP programs as Java code less JSP programs.
- \* Developing custom JSP tags and tag libraries is complex and time consuming process.
- + To overcome above two limitations and to make JSP programs as Java codeless JSP programs use third party supplied <sup>ready made</sup> custom JSP tag libraries like JSTL, display tags, struts tag libraries and etc.

#### JSTL

- Given by Sun micro systems
- is not part of JSP technology
- should configure and should be used with JSP programs separately.
- Gives 4 no. of JSP tag libraries. They are
  - (1) Core JSP tag library
  - (2) SQL JSP tag library
  - (3) XML JSP tag library
  - (4) Formatting JSP tag library

- + By using JSP tags of these four JSP tag libraries we can perform all operations using tags like variable declaration, write data to browser window, conditional operations, Iterational operations, Arithmetic and logical operations (also take the support of EL), DB interactions, XML processing, Formatting data and etc..
- \* Since JSTL gives readily available custom JSP tag libraries there is no need of designing JSP tags, developing tag handler classes and tld files.
- + Core tag library is given for basic operations like variable declarations, flow of control, conditions and iterations.
- + SQL tag library is given for DB interactions.
- \* XML tag library is given for XML processing.

<u>Tag library</u>	<u>tld file name</u>	<u>recommended prefix</u>	<u>jar files</u>
core	c.tld	c	jstl.jar, standard.jar, saxpath.jar, jaxen-full.jar
sql	sql.tld	sql	"
xml	x.tld	x	"
fmt	fmt.tld	fmt	"

\* Fmt tag library is given for specifying Date formats, Time formats, currency symbols, NumberFormats and etc...

\* For tags of various JSTL tag libraries refer Page nos 116 to 119.

\* we can gather the above given four jar files JSTL.jar, Standard.jar, saxpath.jar, jaxen-full.jar from java.sun.com or oracle.com website.

#### Procedure to work with JSTL tags in java web application

(1) Gather the above said four jar files, keep them in WEB-INF\lib folder

(2) Extract the tld file from standard.jar file and place in webapplication like WEB-INF folder.

(3) Read the documentation about JSTL tags

(4) Configure JSTL supplied tag libraries in web.xml file having taglib uris.

(5) Use JSTL tags the jsp programs of web application.

\* while working with JSTL we can use more implicit object along with the regular 9 implicit objects. They are

(1) requestScope

(2) param

(3) paramValues

(4) header

(5) headerValues

(6) cookie

(7) sessionScope and etc...

\* Tags are not there in tags can not be given to perform arithmetic operations in our JSP programs. For this purpose we can take the support of EL which can be used with tags or without tags.

### Syntax

\$ {---- <ELCode> ----}

\* For example application on JSTL tags refer examples given in 120-122 of Booklet.

\* The implicit object param can be used to gather single request parameter value, whereas the implicit object paramValues can be used to gather all request parameter values.

### Web Sphere Server

Type: Application server software

version: 6.0 (compatible with jdk 1.4)

Vendor: IBM

default port no: 9080 for web applications

Commercial software

JAR file that represents JEE : j2ee.jar

download s/w as ZIP file from www.IBM.com website.

Allows to create domains.

If websphere s/w is installed as windows service then each domain must be started services of control panel.

\* To install websphere 6.0 use install.exe file of WAS folder that comes by extracting IBM websphere Application Server v6.0 win rar/ zip file.

\* To create domain in web sphere server

Start → Programs → IBM Web Sphere → Application Server v6 → Profile creation wizard.....

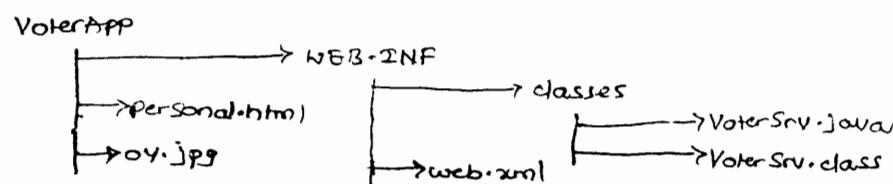
\* In websphere server war file based console deployment is possible.

### Procedure to deploy Java web application in websphere 6.0

Step(1): Set path to jdk 1.4 that comes with websphere s/w installation.

> PATH = D:\Program Files\IBM\AppServer\java\bin

Step(2): Prepare war file on deployment directory structure of web application.



NOTE: compile VoterSrv.java in jdk1.4 environment.

<!DOCTYPE ...> statement in web.xml file is mandatory for websphere server.

>jar cf VoterApp.war .

step(3): start the domain server of websphere from services of control panel.

control panel → performance and maintenance → administrative tools → Services → IBM Websphere Server → start.

step(4): open administrative console of domain server.

start → IB Programs → IBM websphere → application server v6 → profiles → mydomain → Administrative console → userID: xyz  
↓  
userdefined.

step(5): Deploy the web application

Admin console → Install new application → Browse and select VoterApp.war file → context root: VoterAPP → next → next → Continue → next → next → next → Finish → Save to master configuration → save.

step(6): keep the deployed VoterApp application in running mode.

Admin console → Enterprise applications →  VoterApp.war → start

step(7): Test the web application.

open browser window → <http://localhost:9080/VoterApp/person.html>

what is the difference between encodeURL(-), encodeRedirectURL(-) methods of response object ?

(a) encodeURL(-) is useful to append session id to the given URL. This is useful while working with HttpSession with URL rewriting technique.

encodeRedirectURL (-) is useful to send session id from one web application to another application then they can interact with each other with help of sendRedirect (-). This method is useful to make session obj of one web application to another webapplication.

Example on encodeRedirectURL (-) (In SRV1 pg of WAI webapplication)

```
String enurl = res.encodeRedirectURL("http://machine20:7001/WA2/SRV2");  
c.s.sendRedirect(enurl);
```

The above code sends session id from WAI web app to WA2 web app

## Annotations

- \* Data about data is called as meta data.
- \* Gathering more details about already given details is called meta data operation.
- \* In JDBC level we can perform meta data operations in 3 modes.
  - (1) Database meta data
  - (2) Parameter meta data
  - (3) ResultSet meta data
- \* Configuration of resource programs in xml files comes under meta data operations.  
Ex: By configuring Servlet, JSP programs in web.xml we are passing more details about those programs to underlying container or server software. This is nothing but meta data operations.
- \* While working with advanced technologies we generally use xml files to perform resources configuration based meta data operations.
- \* Annotations are the Java statements and they are alternate for the xml files based meta data, resources configuration operations.

### Syntax:

```
@ <annotation name> (param1=val1, param2=val2, -----)
```

- \* To read and process XML documents we need XML parser which is a heavy weight software so it may kill the performance of the application but XML files give good flexibility of modification without disturbing the Java source code.
- \* Annotations will be recognized by Java tools or underlying containers automatically so they give good performance but they give bad flexibility of modification.

### Two types of annotations

- (1) Documentation related Annotations (should be used in /\* \*--\*/ comments)
- (2) Programming related Annotations (can be used in Java source code level)
  - (1) → ex: @author, @param, @see, @since, @return ...
  - (2) → ex: @Override, @WebServlet, @ServletFilter, @InitParam and etc..

- \* Annotations for documentation are given from jdk 1.1 onwards (refer src.zip file of JDK SW).
- \* Annotations for programming are given from jdk 1.5 so all technologies that are comparable with jdk 1.5 gives support for annotations.  
Ex: EJB 3.x, Struts 2.x, Spring 2.5, 3.x, GJB 3.3+, servlet 2.5, servlet 3.x and etc.
- \* Each annotation is a special java interface (with @interface keyword) and parameters of annotation will be declared as java methods.
- \* Parameters of annotations are like attributes of xml tags that means they can be used to configure additional details about resource.
- \* Recognizing annotations and recognizing annotation related resources and applying functionality on them is the responsibility of underlying jdk or jre or container software.
- \* we can apply annotations at 3 levels.

- (1) Resource level (class level / Interface level)
- (2) Method level
- (3) Field level (Member variable level)

\* Annotations helps the programmer to configure the resources as required for underlying Container Software or framework software.

Setup required to servlet 3.0 based servlet programs (no web.xml file)

JDK 1.6 + (Jdk 1.6)

Tomcat 7.0+ (The server that is comparable with servlet 3.x specification).

GlassFish 3.x

Important annotations of Servlet 3.x programming

- @ WebServlet → to configure servlet program with logical name and url pattern.
- @ InitParam → to configure init parameters of servlet program.
- @ ServletFilter → to configure servlet filter program.
- @ WebServletContextListener → to configure ServletContextListeners.

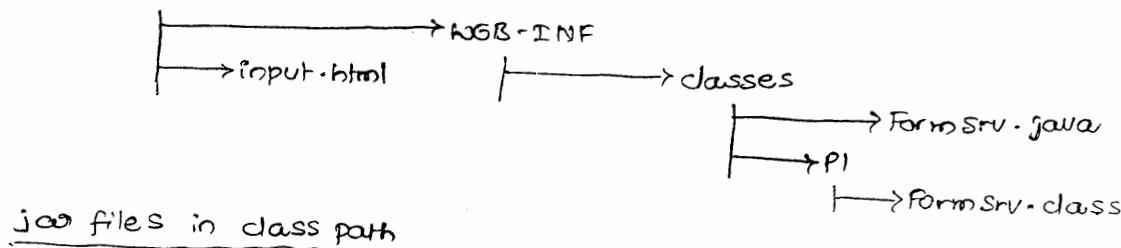
;

\* All these annotations are part of servlet 3.x api and can be collected from  
E:\ncat.home(7)\lib\servlet-api.jar.

- \* The main feature of servlet 3.x is good support for annotations and ability to develop servlet programs based web application without web.xml file.
- \* From JDK 1.5 onwards each java package contains classes, interfaces, annotations and enums.
- \* The servlet api 3.0 is given having 3 packages.
  - javax.servlet, javax.servlet.http and javax.servlet.annotation.
- \* To get servlet 3.x api documentation refer Sunwees api docs..

### Example application (using servlet 3.x annotations)

#### FormAnnoApp



<Tomcat 7-home>\lib\servlet-api.jar file

Deploy above application in Tomcat 7.0 or GlassFish 3.x server

#### input.html

```

<form action="test1">
  Name: <input type="text" name="username"> <br>
  <input type="submit" value="check" >
</form>
  
```

#### FormSrv.java

```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet
@WebServlet(name = "abc", urlPatterns = {"/test1"})
public class FormSrv extends HttpServlet {
    public void doGet(--) throws SE, IOE {
        String username = request.getParameter("username");
        response.setContentType("text/html");
    }
}
  
```

```

PrintWriter pw = response.getWriter();
pw.println("Hello Mr." + username + " Today's Date :: " + new java.util.Date());
pw.close();
}

public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doGet(request, response);
}

```

Request URL

<http://localhost:2525/FormAnnotationApp/input.html>

- \* For related info on servlet 3.0 annotation refer supplementary handout given on 18-12-2011.
- \* NetBeans 6.9+ is giving support for annotations based servlet programming.
- \* Reflection means mirror image.
- \* Reflection API application is capable of gathering given Java class or interface details programmatically and dynamically.
- \* Being from one application if you want to gather certain Java class details like member variables, methods and etc. details programmatically then develop the code of that application by using reflection API.
- \* To develop our own annotations we need to develop one simulator class called container on the top of JRE then only we can provide functionality to that annotation.

Procedure to develop user defined annotation having the following functionality

@sum(num1=10, num2=20)

int result → should assign 30 to result variable.

@sum(num1=20, num2=40)

int res → should assign 60 to res variable.

Step(1): make sure that JDK 1.5+ software is available.

Step(2): Design your annotation.

package P3;

import java.lang.annotation.\*;

Retention(RetentionPolicy.RUNTIME) makes annotation as runtime annotation

@Target(ElementType.FIELD) *This annotation is applicable only on fields.*

```
public @interface Sum {  
    int num1() default 0;  
    int num2() default 0; } parameters.
```

↳

Step(3): develop your own container class.

```
package P3;  
import java.lang.reflect.Field;  
public class Container {  
    public static void addThis(Object theObject)  
    {  
        try {  
            //get access to all the member variables of current obj (Test class obj)  
            Field[] fields = Class.forName(theObject.getClass().getName()).getDeclaredFields();  
            for (int i=0; i< fields.length; i++)  
            {  
                //get access to @Sum annotation  
                Sum obj = fields[i].getAnnotation(Sum.class);  
                if (obj!=null)  
                {  
                    //getting values from annotations parameter  
                    int val1 = obj.num1();  
                    int val2 = obj.num2(); int tot = val1+val2;  
                    //setting it to our class variable (like result)  
                    fields[i].setInt(theObject, tot);  
                } //if  
            } //for  
        } catch (Exception e) { e.printStackTrace(); }  
    } //addThis
```

Step(4): use the annotation in your application

```
package P3;  
public class Test {  
    @Sum (num1=15, num2=10)  
    static int result;  
    public static void main(String args[]){  
        Test t1 = new Test();  
        Container.addThis(t1); //activate user-defined container class  
        System.out.println("The sum is:" + result);  
    } //main
```

Step 5: compile and execute the application.

>javac -d . \*.java (Compilation)

>java P3.Test (Execution)

technicalnumbo.wordpress.com

\* There are multiple ways to know the details of java class or interface.

They are

- (1) By working with javap tool
- (2) By working with java api documentation.
- (3) By opening source code of java resources.

javap :

\* It is a built-in tool of jdk software.

Ex: javap java.lang.String  
javap java.io.Serializable  
javap MyTest  
---

\* In jdk software installation we can use java\_home\src.zip file to get the source code of all the built-in classes and interfaces of java api.

+ When all the above three techniques are used to gather details about certain java class or interface we can not use those details as input values in our java application.

\* Reflection means mirror image.

\* Reflection api based java applications can gather the internal details about given java class or interface programmatically and dynamically and these details can be used in the application as input values for other requirements.

\* Each reflection api based java application acts as mirror image having the ability to gather details about given class or interface dynamically.

\* Reflection api means working with classes and interfaces of java.lang.reflect package.

\* java.lang is default package in java applications but its sub packages are not default packages.

```
public final class Test
{
    float static int a;
    private float b;

    Test()
    {
        =
    }
}
```

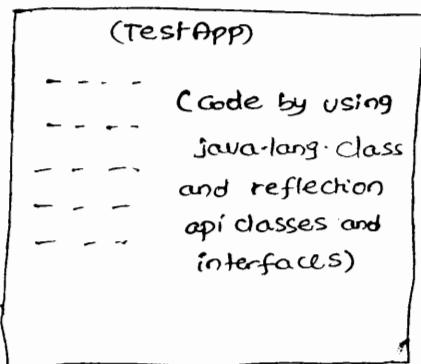
```

Test (int x)
{
    =
}
public int sum (int x, int y)
{
    =
}
}

```

- \* In Java environment member variables are technically called as fields and constructors are called as creation methods.

### Reflection API based App



>java TestApp.java

>java TestApp java.lang.System → command line argument value

→ prints all the details of java.lang.System class as output

>java TestApp java.lang.Runnable

→ prints all the details of java.lang.Runnable (I) as output

- \* Reflection API is popularly used in the development of Java based software products like debuggers, ClassBrowser, GUI Builders and etc....

- \* Debugger allows us to trace flow of execution in the application.

• ClassBrowser is nothing but hint box that comes in IDE s/w

- \* GUI Builder allows the programmer to design the application through drag and drop operations.

• Every IDE contains Debugger, ClassBrowser, GUI Builder products. This indicates the IDE s/w internally using reflection API support.

- \* We can gather the basic details about given class or interface by using various methods of java.lang.Class but use classes and interfaces of java.lang.reflection API to gather advanced details.

## java.lang.reflect package

classes

- Field
- Constructor
- Method
- Modifier

:

Interfaces

- Invocation Handler
- Member

write a java application to gather super class name of given java class

java.lang.Object

↑ extends

java.lang.String

ABC

↑ extends

XYZ

// App1.java

public class App1

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

{ // load given java class into App

Class c = Class.forName (args[0]);

// get super class of given class

Class sc = c.getSuperclass();

// print given class, super class name

System.out.println ("given class name: " + c.getName());

System.out.println ("super class name: " + sc.getName());

} // main

} // class

> javac App1.java

> java App1 java.lang.String → command line argument value

> java App1 java.lang.System

\* when the object of java.lang.Class points to class or interface and to get that class name or interface name as string value use getName() method call on java.lang.Class object.

write a java application to get all the classes of inheritance hierarchy

for a given java class

```

java.lang.Object <|-->
    |
    +-- ABC
        |
        +-- XYZ
            |
            +-- Demo
                |
                +-- Test

```

//APP2.java

```

public class APP2
{
    public static void main (String args[])
        throws Exception
    {
        //load given java class into APP
        Class c = Class.forName(args[0])
        //get Super class
        Class sc = c.getSuperclass();
        // print all the classes of inheritance hierarchy
        System.out.println("classes in the inheritance hierarchy for " + args[0] +
                           " class are:");
        while (sc != null)
        {
            System.out.println("\t\t" + sc.getName());
            c = sc;
            sc = c.getSuperclass();
        }
    }
}

```

}//main

}//class.

//javac APP2.java

//java APP2 java.lang.Integer

//java APP2 java.applet.Applet

Object of java.lang.Class can represent a concrete class or interface or abstract class or data type.

write a java application to get list of java interfaces implemented by the given java class

```

public class Test implements X, Y, Z
{
}

```

5 =

```

//APP3.java
public class APP3
{
    public static void main(String args[])
        throws Exception
    {
        //load the given java class into APP
        Class c = Class.forName(args[0]);
        //call getInterfaces() of java.lang.Class
        Class[] inter[] = c.getInterfaces();
        //print interfaces names
        for(int i=0; i<inter.length; i++)
        {
            System.out.println("//" + inter[i].getName());
        }
    }
}

//javac APP3.java
//java APP3 java.lang.String
//java APP3 java.lang.Thread

* int[] means all some elements in that array are int values.
* java.lang.Class object[] means all elements in that array are the
  objects of java.lang.Class.

* public, final, abstract are the modifiers of java class. But we can not
  apply final and abstract modifiers at a time on a java class because
  final class does not support inheritance and abstract class needs the support
  of inheritance (we can not write subclass to final class but we need to
  write subclass to abstract class to implement abstract method's).

```

write a Java application to get the modifiers of given Java class

```

//APP4.java, import java.lang.reflect.*;
public class APP4
{
    public static void main(String args[])
        throws Exception
    {
        //load the given java class into App
        Class c = Class.forName(args[0]);
        //call getModifiers() of java.lang.Class
        int x = c.getModifiers(); → returns int value representing the modifiers of
                                     given Java class.
    }
}

```

```

//print modifiers
if (Modifier.isPublic(x))
    System.out.println("It public");
if (Modifier.isAbstract(x))
    System.out.println("It abstract");
if (Modifier.isFinal(x))
    System.out.println("It final");

}//main
}//class

//>javac App4.java
//>java App4 java.lang.String
//>java App4 java.lang.Thread

```

- \* `isXXX()` methods of `Modifier` class are boolean methods - checking whether the given numeric value represents certain modifier or not.
- \* By default all interfaces are abstract.
- \* In Java there is no terminology called access specifier.
- \* `public`, `private`, `protected`, `final`, `static`, `synchronized`, `Volatile`, `transient` and etc.. keywords are called modifiers in Java.
- \* The object of `java.lang.reflect.Field` class can represent one member variable of given Java class.

write a Java application to get the details of member variables/ fields available in Java class

```

//App5.java
public class APP5
{
    public static void main (String args[]) throws Exception
    {
        //load the given Java class into App
        Class c = Class.forName(args[0]);
        //call getDeclaredFields() of java.lang.Class
        Field f[] = c.getDeclaredFields();
        //Print Field details
        for (int i=0; i < f.length; ++i)
        {
            int x = f[i].getModifiers(); //gives modifier of each field
            String type = f[i].getType().getName(); //gives datatype of each field
        }
    }
}

```

```

        String name = f[i].getName(); // gives name of each field

        if (Modifier.isPublic(x))
            System.out.println("public");
        ----
        ----
        System.out.println(type + " " + name);

    } // for

} // main

} // class

//> javac APP5.java
//> java APP5 java.lang.String

```

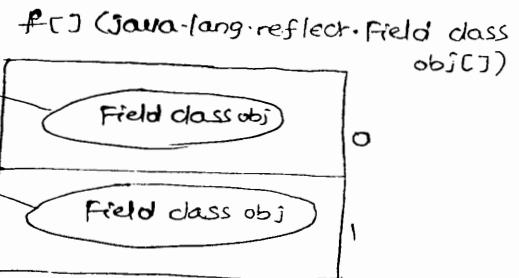
Ex: public class Test

```

{
    private int a;
    static final float b;
}

```

--- methods



Write a Java application to gather all the method details of given Java class

//APP6.java

```

import java.lang.reflect.*;
public class APP6
{
    public static void main (String args[])
        throws Exception
    {
        // load the given Java class into App
        Class c = Class.forName(args[0]);
        Method m[] = c.getDeclaredMethods();
        // print method details
        for (int i=0; i<m.length; ++i)
        {
            int x = m[i].getModifiers(); // gives modifiers of each method
            String rtype = m[i].getReturnType().getName(); // gives return type
            String mname = m[i].getName(); // gives method name
            Class p[] = m[i].getParameterTypes(); // gives parameter types
            if (Modifier.isPublic(x))
                S.o.p("public");
        }
    }
}

```

```

if(modifier.isFinal(x))
    s.o.p("final");
-----
s.o.p(rtype + " " + mname + "(");
for(int k=0; k<p.length; ++k)
{
    s.o.p(p[k].getName() + " ");
}
//for
System.out.println("");
}
//for
}
//main
}
//class
//javac APPG.java
//java APPG java.lang.String
//java APPG java.lang.Runnable

```

### Memory diagram

```

public class Test
{
    public void hello()
    {
        System.out.println("Hello");
    }

    public static final int sum(int x, int y)
    {
        return x+y;
    }
}

```

