




**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** Visit

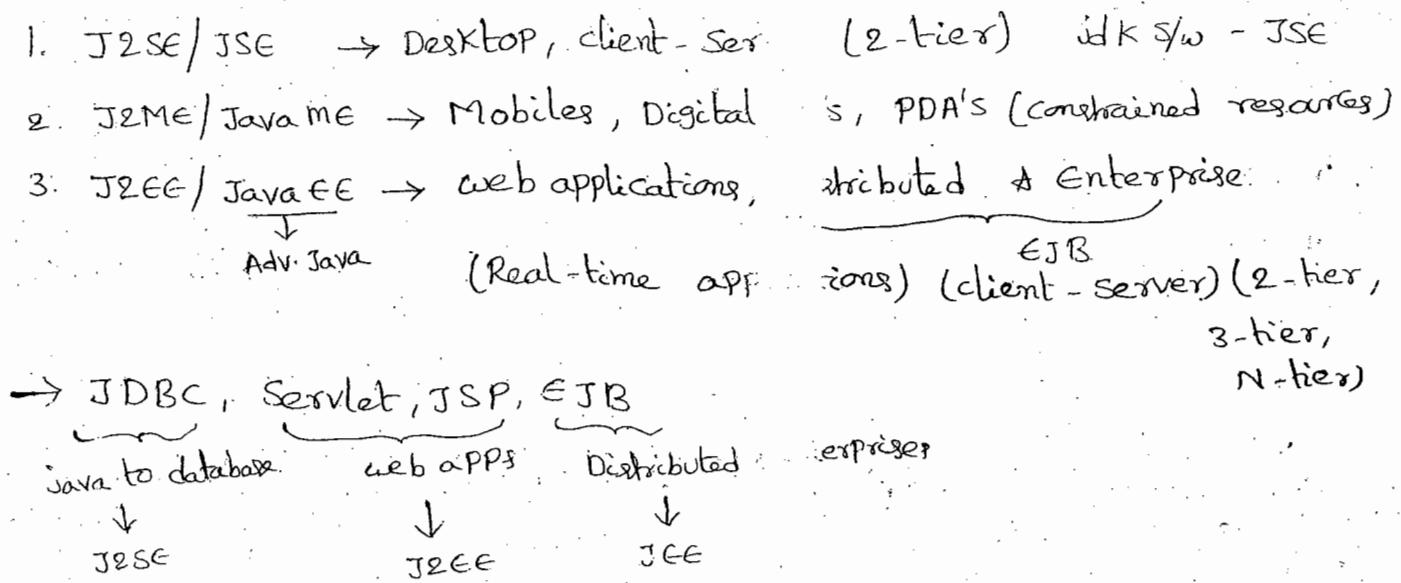
www.tutorial4us.com

A Perfect Place for All **Tutorials** Resources

Advance Java

(Natraj Notes)

www.tutorial4us.com

Introduction:-

"SUN" Microsystems has divided java technology into 3 modules.

1. Java SE (Standard edition)

2. Java ME (Micro edition)

3. Java EE (Enterprise Edition)

1. Java SE :-

JSE is suitable for 2-tier (client-server) applications.

* Java SE module is for developing applications in Java.

* JDBC - JSE.

Java SE is suitable for stand-alone i.e., desktop & standalone. Java SE is an installable software. It supports both console and windows.

Standalone

2. Java ME :-

JME is suitable for resource constrained devices.

Ex:- For mobile applications, PC Digital set-top boxes etc.

The development of app.'s

(Personal Digital Assistants),

JavaEE is suitable for the development of web-applications, Distributed & Enterprise applications.

* JavaEE is not an installable S/w.

* Javase is the base for JavaEE

Servlet, JSP

* JavaEE is a S/w in the form of jar (java Archive) files

* JavaEE is totally for the development of client-server applications only 1-tier, 2-tier, 3-tier & n-tier

* Web applications are servlet, JSP

* Distributed & Enterprise applications are: EJB

* In JavaEE, we have the following 3 technologies.

1. Servlet

2. JSP

3. EJB

* Servlet, JSP technologies are for development of web applications.

* EJB technology is used for development of distributed & enterprise application.

Client-Server models :-

client-server model applications by depending on the 3 logics of the project.

1. presentation logic/layer

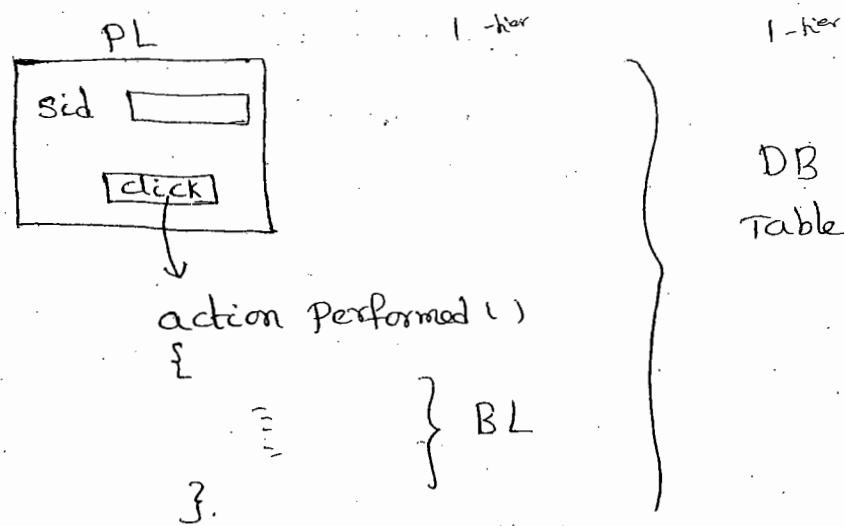
2. Business logic

3. Data Access logic/persistence logic.

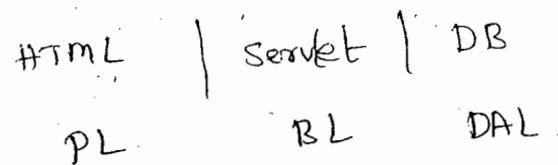
Model :-

Model means an application (or) software.

- To develop client-server applications we do not require physically multiple systems as machines.
- In a single physical machine, we can able to develop entire applications also.
- The client-server model will be decided either by using the no. of S/w's communicating or no. of application communicating, but not no. of systems communicating.
- Single-tier applications are nothing but the applications that contains all the 3 logics inside it.
ex :- MS-Access
- In client-server model applications, depends on the logic, the applications are consider as a two-tier (or) 3-tier (or) n-tier.
- In advance java, we use JDBC, servlet, JSP technologies
- JDBC technology is for the development of 2-tier.



- servlet & JSP technologies are 3-tier & n-tier applications.



Web

A web application is a server side application. It runs at server or executes at server and provides service to the clients across internet.

→ Web applications produce web pages onto the client browser.

For example, every web site is a web application.

→ Generally, websites are providing services like mail and chatting.

For example, the websites like gmail & yahoo are providing both mailing & chatting services to the clients across the n/w.

→ Whenever, we want to provide service to multiple clients across internet then we need to choose web apps.

→ If at all we want to provide service to a single client at a time. Then we need to choose a desktop applications.

→ In today's world, most of the web application development is done by using Java technologies like Servlets, JSP's

→ The difference b/w a web application & a website

is a web application is an application that has not yet placed into an internet server. But once, it is placed in an internet server then it becomes as a website.

Q What is the difference between S/w project & S/w product?

Ans:- 1. If an application is developed, according to the requirements of a client organization then we call that application as a S/w project.

2. If an application is developed, not according to

The requirements of an organization otherwise client organization then we call that application as a s/w product

* For example, Finacle this is the client for Infosys.

Finacle is a s/w project developed by Infosys for a client called ICICI.

* Oracle is a s/w product and it can be used by any time.

→ web applications are divided into 2 types:

1. presentation oriented

2. service oriented

→ To develop presentation oriented web applications we do not require any technologies. Just HTML pages are enough.

For example, online tutorials or presentation oriented web applications.

↓
Servlet, JSP. www.roseindia.net

→ Service-oriented applications development uses web technologies. These type of applications are concentrating on providing services to the client.

for example, gmail, yahoo, rediff.

→ To develop service-oriented web applications, so far we got the following technologies.

(i) CGI (common gateway interface)

(ii) Servlet

(iii) Server pages

→ Java developers uses Servlet & JSP technologies & .Net people uses asp.net for the development of service-oriented web applications.

minima

Reflection is a mechanism used for finding reflective information about a class or an interface of java.

- Reflective information is nothing but getting metadata of a class or an interface.
- Meta data of a class means, finding the details about the class like its base class, interfaces implemented by the class, constructors of the class, Methods of the class & variables of the class (fields).
- Reflection is used for work finding metadata of a pre-defined class or metadata of a user-defined class also.
- we can find metadata of the current class which is running or we can find metadata of another class, by loading it.
- Reflection is required for the programmers who are interested in the development of s/w products.
- Reflection is used mostly while developing java classes, browsers, java debuggers & java compilers.
- Reflection is internally used in the development of IDE's.

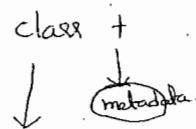
Loading a class:-

whenever, we want to find details (metadata) of a class or an interface then first of all that class or interface must be loaded into JVM.

* In order to load a class or an interface into JVM at run-time (dynamic), we need to use a method forName().

forName() :-

forName() is a static method even in class.
So, we need to call forName() method, with respect to
class name. class +



- whenever a class is loaded into JVM then internally JVM separates metadata into an object and then that object will be return back to the programmer.

- If a programmer wants to know the metadata, then the programmer receive the object & otherwise no need to receive the object.

- while loading a class, first JVM verifies whether the class is already loaded or not. If not loaded then it loads the class now. If it is already loaded then the class is not loaded once again.

- A JVM loads a class for only one

class → Keyword.

class → class name (java.lang)

↓ already defined.

class A

۹۷

P.S.V.M()

۳

class C = class.forName("B")
 ↓ ↓ ↓
 Object static method

3

YI C++

ERROR: Could not find String

↳ fully qualified class (including package)

C:\> javap java.lang.String <

= } metadata
=

Ex:- public class st extends String ^{final} ↗ final
↓ doesn't extend.

→ we can also find metadata of a class or an interface

using a tool given by JDK or Javap (Java profiler).

→ For example, If we want to know metadata of String class
then the following command is required.

C:\> javap java.lang.String

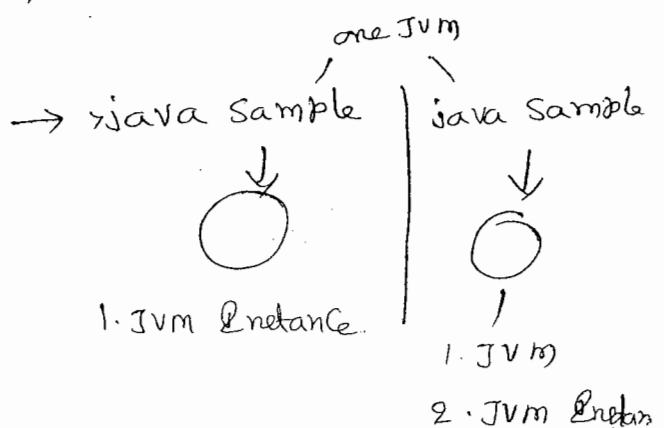
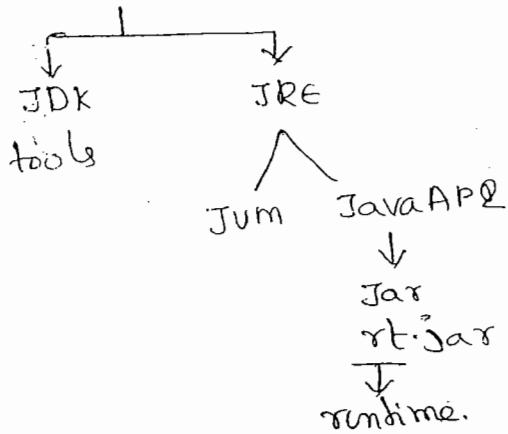
→ Reflection mechanism = Javap tool. But the difference is,
Javap is only used at command prompt, but not written a
java application.

→ the basic step to enter into the reflection programming is:
loading a class. This can be done by using the following

Syntax:

```
class c = Class.forName("fully qualified class name");
```

→ Java S/w



for ex

class c = class.forName("java.lang.Integer");

wrapper class

Q. what is a factory method?

Ans A method which produces an object is called as an factory method.

* Factory methods are of 2 types:

1. static factory methods

2. instance factory methods

* The following statements are examples for static factory methods in java.

1. class c = class.forName("java.lang.String");

2. Calender c = Calender.getInstance();

↓
abstract class & interface not create objects

3. Thread t = Thread.currentThread();

Q. In how many ways dynamic loading of a Java class into Jvm is possible?

Ans: 2 ways

1. By using forName() method given by Class c.

2. By using loadClass() method given by ClassLoader.

class c = class.forName("fqcn")
↳ fully qualified class name

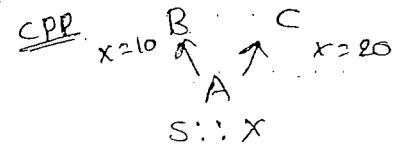
class c = classLoader.loadClass("fqcn");

Note:- whenever a class is loaded into jvm then not only that class, but also its related classes & interfaces are also automatically loaded into jvm.

~~Following super-class, base class~~

1. Whenever a class is loaded into JVM then automatically its superclass is loaded into JVM.
 2. If we want to get superclass metadata then we need to call a method `getSuperclass()`.
 3. In Java, one class can extend a maximum of one base class at a time. The reason is Java doesn't support multiple inheritance at class level.
- Q. Why Java doesn't support multiple inheritance at class level?

Ans:- If multiple inheritance is supported then there is a chance of getting ambiguity error. To solve this error, we need scope resolution (::) operator. But this operator is not supported in Java. So multiple inheritance is not supported in Java at class level.



Note:- At interface level, Java supports multiple inheritance. But in an interface all variables are static & final. So, no need of scope resolution operator.

→ The following code is to print `superClassName` for the loaded class.

```
class c = Class.forName("java.lang.Integer");
Class sc = c.getSuperclass();
String str = sc.getName();
System.out.println(str);
```

→ The following example is for printing superclasses name for the given class as a command line argument.

// Reflect1.java

```
class Reflect1  
{  
    static public void main(String[] args)  
        JVMM will  
        handle → throws Exception  
    {
```

```
        class.forName(java.util.String)  
        class c = Class.forName(args[0]);  
        class sc = c.getSuperclass();  
        String s1 = c.getName();  
        String s2 = sc.getName();  
        System.out.println(s1 + " extends " + s2);  
    }
```

?; // optional

O/P:-

→ javac Reflect1.java

→ java Reflect1 java.lang.Integer ↴

java.lang.Integer extends java.lang.Number.

→ java Reflect1 java.lang.Exception ↴

java.lang.Exception extends java.lang.Throwable.

→ In the above example, we have uses throws exception statement.

For the main method. The reason is forName() method.

throws ClassNotFoundException if the given class is not found.

→ In Java end of the class can be represented with semicolon (;) But it is optional.

It is given by JDK.
String... args.
(or)

we can also write args
because it array var

class.forName(java.util.String)
cause exception
because we can use big

- whenever a class is loaded into jvm then automatically all implemented interfaces are also loaded.
- If we want to find metadata i.e., the interface names implemented by the class, then we need to call a method called getInterfaces().
- The method getInterfaces() returns an array or of class type.

By using a loop we can get name of each interface.

→ The following example is for finding the interface names implemented by a class, which is given as command line argument

```
// Reflect2.java
```

```
class Reflect2 {
    public static void main (String [] args) throws exception
    {
        Class c = Class.forName(args[0]);
        Class it[] = c.getInterfaces();
        for (int i=0; i < it.length; i++)
        {
            String s1 = it[i].getName();
            System.out.println(s1);
        }
    }
}
```

Op:- c:\>javac Reflect2.java

c:\>java Reflect2 java.lang.String

java.io.Serializable

java.lang.Comparable

java.lang.CharSequence

c:\>java Reflect2 java.lang.Thread

java.io.Runnable ← interface

Q :- Difference b/w length and length() :

- Ans :-
1. length property is used for finding size of an array
2. length() method is used for finding for size of a string.

For example:- String str[] = {"Java", "oracle", "CPP"};
str.length
str[i].length()

Q In Java, why we are importing parent & child packages separately?

Ans:- Because, at Packages level java doesn't support inheritance so, we are importing parent & child packages separately in java.

class extends class

interface extends interface

class implements interface

package package (Not supported)

Note:- Reflection always getting data from heap (instance variable stored (or) fields store)

Finding fields data:-

1. A field is nothing but an instance variable.
It means a variable created (or) declared inside the class.
2. In Java, we can also declare variables inside a method (or) block. we call those variables as "local variables".
3. Reflection can be used to find the variables details that are created inside a class. we call these variables as "fields".
4. Reflection doesn't provide details of local variables because, local variables are stored in "stack memory".

```

    {
        int x; → field (class scope)
        void m()
        {
            int y; → local variable (method scope)
            if (...)
            {
                int z; → local variable (block scope)
            }
        }
    }

```

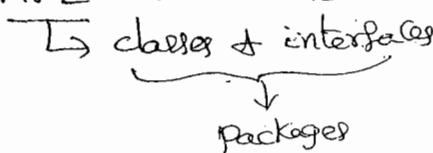
→ while finding details of fields, we need to store the each field metadata into an object of type Field class.

→ Field is a class given in java.lang.reflect package.

→ java.lang.reflect is a subpackage of java.lang (Default Package)

→ Reflection API contains a single package called java.lang.reflect.

In Java API is nothing but set of classes & interfaces.



→ java → API → classes & interfaces → Packages

c++ → API → Functions & classes → header files

c → API → Functions → header files

Java 1.5 - 3562 classes

1.6 - 3562 + _____ classes

Note:- when we install java s/w, we will get the java API into our system in the form of "rt.jar" (runtime)

→ In Reflection, we have to import "java.lang.reflect" package explicitly into a java program. the reason is, if a parent package is imported, then child package is not automatically imported

The following example is for finding the details of the fields of a loader class.

//Reflect3.java

```
import java.lang.reflect.*;  
  
class Reflect3  
{  
    public void m (String [] args) throws exception  
    {  
        //Handle Exception  
        class c = Class.forName(args[0]);  
        field f[] = c.getFields();  
        for(int i=0; i < f.length; i++)  
        {  
            String fname = f[i].getName();  
            String dname = f[i].getTYPE().getName();  
            S.O.P(dname + " " + fname);  
        }  
    }  
}
```

O/P:

```
c:\> javac Reflect3.java  
c:\> java Reflect3 java.lang.Thread  
int MIN-PRIORITY  
int NORM-PRIORITY  
int MAX-PRIORITY
```

Q:- what is a singleton variable in Java?

Ans:- If a variable is public, static & final then.

that variable is called "singleton variable"

Q:- what is singleton class?

Ans:- For a class, if it is possible to create only one object then we call that class as a "one and only" class.

Ans: Yes

example:

```
class Sample  
{  
    static  
    {  
        String str[] = {"abc", "xyz"};  
        main(str);  
        System.exit(0);  
    }  
    p.s.v.m(String[] args)  
    {  
        S.O.P ("I am main()...");  
    }  
}
```

c:\> javac Sample.java

c:\> java Sample

I am main()... } called by programmer

Q what is the difference between getFields() method & getDeclaredFields() method?

Ans: getFields() method returns the metadata of derived class fields including its base class fields also. But, getDeclaredFields() method returns only metadata of the fields related to the current class only, but not its base class.

Actually, main() is called by JVM but here we have to call the main() with out JVM

- Once a class is loaded, if we want to find the details of the constructors of a loaded class then, we need to call getConstructors()
- For each constructor metadata, an object of type Constructor class is created.
- Constructor is a class given in "java.lang.reflect" package.

```
Class c = Class.forName(" ");
Constructor cons[] = c.getConstructors();
```

Note:- Constructors doesn't have return type but method has a return type. By default constructors can be created at compile-time.

Note:- Constructors can't be inherited, but called from base class. Methods, variables can be inherited from base class. Constructors can be executed from top to bottom and destructors can be executed from bottom to top and in Java, we use finalize() instead of Destructors.

Example:- The following example is for printing details of constructors of a loaded class.

// Reflect4.java

```
import java.lang.reflect.*;
class Reflect4
{
    public static void main (String [] args) throws Exception
    {
        Class c = Class.forName(args[0]);
        Constructor cons[] = c.getConstructors();
        for (int i=0; i< cons.length; i++)
        {
            String conname = cons[i].getName();
            System.out.print(conname + "(");
        }
    }
}
```

class type of \leftarrow to get the parameters
 an array of the constructor.

```

for (int i=0; i<p.length; i++)
{
  String ptype = p[i].getName();
  s.o.println(ptype + " ");
}
} //inner
System.out.println("j");
} //outer
}
}
  
```

O/P:-

```

c:\> javac Reflect4.java
c:\> java Reflect java.lang.String
java.lang.String([C, int, int])
  
```

Note:- Java.lang.Math & Java.lang.System, both classes

contains private constructors in the class. so, we cannot

create an object of Math & System classes directly.

* All methods are in Math & System classes are static, because

we call by using class name

Math.Pow(a,b)

System.out.println();

↓
obj of PrintStream class
created in System class.

Note:- Math and System classes contains all methods and variables as static. so, we can call the methods directly by using class name.

Note:- static methods doesn't allow this and super

Keywords of Java.

* static methods allows only static variables of the class
but instance variables are not allowed.

* If variables declared inside the static, it is local &
it can be accessed.

```

class p.s.v.in(string arr)
{
    static int a=50;
    static void print()
    {
        int m=40;
    }
}
  
```

Q Why sizeof() is removed from Java?

Ans:- In Java, memory for all primitive types are constant
and doesn't change from one O.S. to another O.S. But,
In C & C++, according to the O.S., memory will be
changed. Because of this reason, sizeof() method is
given (or) available in C & C++ and it is removed from

java.	DOS	32bit	64 bit
C int a;	16 bit 2 bytes	4 bytes	8 bytes → P.D.
Java.	4	4	4. → P.L.

Note:-

Java can execute faster in UNIX O.S than it
executed in any OS. The execution speed depends on
the data transfer.

Finding methods :-

1. Once a class is loaded into JVM, if we want
to find the methods details of the class then we
need to call getMethods().

2. getMethods() returns an array of Method type.

3. Method is a class given in java.lang.reflect.

inherited but constructors cannot be inherited.

5. we can also call `getDeclaredMethods()`, for finding the details of the methods of a loaded class.

*6. The difference between `getMethods()` & `getDeclaredMethods()` is, `getMethods()` means, it returns current class methods and also base class methods. whereas, `getDeclaredMethods()` means, it returns only the current class methods, but not including the base class methods.

7. Finding methods is similar to finding constructors. But, the difference is, for a constructor we do not have any return type. whereas, for a method we need return type also.

The following example is for printing the methods of a given class.

/*Reflect5.java

```
import java.lang.reflect.*;
```

```
class Reflect5
```

```
{
```

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

```
        Class c = Class.forName (args[0]);
```

```
        Method m [] = c.getMethods ();
```

```
        for (int i=0 ; i<m.length ; i++)
```

```
{
```

```
            String rname = m[i].getName ();
```

```
            String rtype = m[i].getReturnType().getName();
```

```
            System.out.print (rtype + " " + rname + "(");
```

```
class PLJ = mLoc.getParametersTypes();
```

```
for (int i=0; i<P.length; i++)
```

```
{
```

```
String ptype = P[i].getName();
```

```
System.out.print(ptype + " ");
```

```
} //inner
```

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

```
} //outer
```

```
}
```

O/P:

```
C:\>javac Reflects.java
```

```
C:\>java Reflects java.lang.Exception
```

```
no methods in Exception class.
```

10/6/11

Finding Modifiers:-

1. In Java, we do not have any access specifiers instead we have all access modifiers.
2. for a class (or) an interface in Java, we cannot provide "private" & "protected" modifiers.
3. A class (or) an interface in Java can be either "public" (or) default".
4. In a Java application (or) program, if there multiple classes then only one class can be made as "public". The public class name and the program name, both must be same.

Note:-

Sample.java

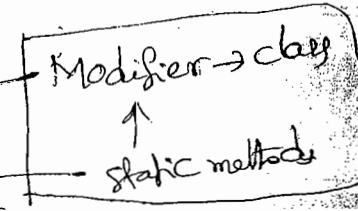
```
class A
{
}
class Demo
{
    main();
}
```

```
javac Sample.java
```

```
java Demo
```

```
java.lang.reflect
```

```
methods are
```



find modifiers of the class then we need to call the method `getModifiers()`.

- getModifiers() method returns an integer value. This integer value indicates whether a class is public (or) final (or) static (or) abstract.

Ex- class c = class.forName(args[0]);
int k = c.getModifiers();

- In `java.lang.reflect` package, we have a class given called "Modifier" and this class provides "static methods" to find whether the int value belongs to which modifier.
 - For example, if we want to check whether a class is public (or) not then, the following condition is used.

```

    if (Modifier.isPublic(k))
    {
        S.O.P("public class");
    }

```

$k \rightarrow$ 1 Public
 \sqcup (public final)
 g [static]
 l [public abstract]

- The following example is for finding modifiers of a class, by taking input as command line argument.

// Reflect6.java

import java.lang.reflect.*; (In real-time code we can't use the
*, we can provide particular class).

class Reflect6

3

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

1

```
class c = class.forName(args[0]);
```

```
int K = c.getModifiers());
```

`if (Modifier.isPublic(k))`

۷

S.O.P ("public class");

```

if (Modifier.isSource(k))
{
    S.O.P ("static class");
}

if (Modifier.isFinal(k))
{
    S.O.P ("final class");
}

if (Modifier.isAbstract(k))
{
    S.O.P ("Abstract class");
}
}
}

```

Note:- It's not compulsory to save the class name as a program name.

O/p:-

```
c:\>javac Reflect6.java
```

```
c:\>java Reflect6 java.lang.String
```

```
public class
```

```
final class
```

```
c:\>java Reflect6 java.util.Calendar
```

```
public class
```

```
Abstract class
```

→ Note:- In Java, abstract and final both modifiers are not allowed for a class at a time. The reason is,

abstract and final are opposite keywords.

* If a class is an abstract, then, that class must be extended (inherited). If a class is final, then that class cannot be extended. So, both keywords cannot be applied at a time.

~~Q~~ what is the difference between an inner class and a Nested class?

static inner class is called "Nested"

Ex:-

class A

{

 class B → inner class

{

}

}

class A

{

 static class B → Nested class

{

}

}

~~Q~~ Can we find size of object in java (or) not.

Ans:- Yes, it is possible. we need to use "Runtime" class of `java.lang.*` package.

For example:-

Jdk 1.4 mechanism

method returns an object
called factory method.

Runtime r = Runtime.getRuntime();

long x = r.totalMemory() - r.freeMemory();

x → It is for storing used memory before creating an object

long y = r.totalMemory() - r.freeMemory();

y → It is for storing used memory after creating an object

long z = y - x;

S.O.P (z + "bytes");

In Jdk 1.5 mechanism, java program execution starts from

premain(). It has two main().

1. premain()

2. main()

The size of an object can be find in jdk 1.5 by using Instrumentation interface. it is in `java.lang.instrument`.

long x = ins.getobjectsize();

Q How many ways we can create an object for a class in Java?

Ans: There are 5 ways to create an object for a class in Java.

1. new keyword
2. newinstance()
3. factory method
4. clone()
5. deserialization()

Note:- Among all the above 5 ways, new keyword is very expensive to create an object for a Java class. So, in Real-time applications, the new keyword is used for less no. of cases.

Explain → When a Java programmer wants to create an object for a class at client side applications. Then mostly new keyword is used.

- new keyword is very expensive, in terms of performance.
- new keyword allocates memory in byte by byte manner.
- newinstance() method is used by servers, while creating objects for the classes.
- The advantage of newinstance() is, class name can be provided at run time dynamically to create an object of the class.

* The difference b/w new keyword & newinstance() is, in a new keyword class name must be known before compiling, to create an object for the class. But in a newinstance(), class name can be given at run time for creating an object of the class.

we can not create an object for the class from outside of the class. In this case, we use factory method for creating an object of the class.

- * Mostly we use factory method, to convert a class into a singleton class.
- * If we want to create more number of objects for the same class, then instead of using new keyword for multiple times, we prefer clone() method.
- * cloning is the process creating a duplicate object for an existing object.
 - * The advantage of clone() is the entire memory will be copied at once. So, it increases the performance of an application.

Ex:- Sample s1 = new Sample();
 Sample s2 = s1.clone();

DeSerialization:-

It is a mechanism used in distributed programming of java.

- * When an object is transferred from one system to another system in a network then the bytes of an object are transferred across n/w. At the ^{other} end these bytes are received and stored into an object. This process is called "DeSerialization".



→ The following example is to create an object for
a class using factory method.

//factoryTest.java

class Sathya

{

 private Sathya()

{

}

 public static Sathya getObject()

{

 return new Sathya();

}

 public void display()

{

 System.out.println("display");

}

}

class factoryTest

{

 public static void main(String[] args)

{

 // Sathya s2 = new Sathya(); error

 Sathya s1 = new Sathya();

 Sathya s1 = Sathya.getObject();

 s1.display();

}

O/P:-

C:\> javac factoryTest.java

C:\> java factoryTest ↴

display

a static factory method.

- * If getObject() method is called for 2 times then 2 times object is created for the class Sathya. So, class Sathya is not a singleton class.
- * In the above example, we can convert the class Sathya into singleton. To do this create an object at outside of the getObject() method.

For example:-

```
class Sathya
{
    static Sathya s=new Sathya();
    return s;           (Continue)
}
```

- * Static method will allow only static variables of the class. Because of this reason, we made 'S' as a static object.

→ The following code (or) example is to create an object by using clone()

//cloneTest.java

```
class Sample implements Cloneable
{
    public Sample()
    {
        System.out.println("Object created");
    }
    public Sample clone() throws CloneNotSupportedException
    {
```

```

        Sample s = (Sample) super.clone();
        return s;
    }

}

class cloneTest
{
    public static void main(String[] args) throws Exception
    {
        Sample s1 = new Sample();
        Sample s2 = s1.clone();
        Sample s3 = s1.clone();
    }
}

```

O/P:- c:\> javac cloneTest.java

c:\> java cloneTest

object created.

Note:- In the above example totally 3 objects are created. But constructor is executed for only once. The reason is, we use clone() method for creating two objects.

* Q Can we create an object for a class, by without calling constructor of the class?

Ans:- Yes, Possible. It can be done by using clone() method.

→ This program is for factory method.

class Sathya

(Contd.) {

static Sathya s = new Sathya();

private Sathya()

10Y

```
{  
}  
public static sathya getobject()  
{  
    return s;  
}  
public void display()  
{  
    System.out.println("display");  
}  
};  
  
class factoryTest  
{  
    public static void main (String [] args)  
{  
        sathya s1 = sathya.getobject();  
        s1.display();  
        sathya s2 = sathya.getobject();  
        s2.display();  
    }  
}
```

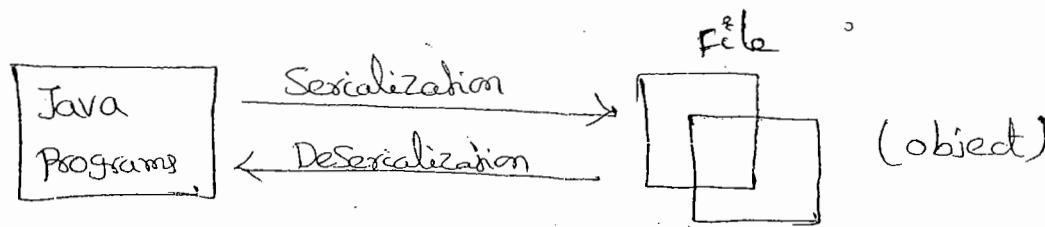
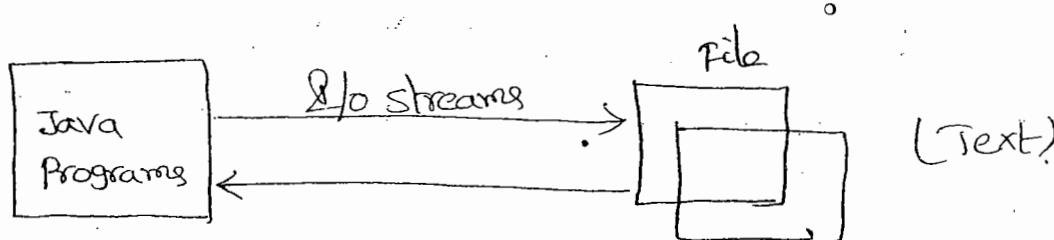
13/10/

Java DataBase Connectivity (JDBC) :-

file Management System (FMS) :-

storing data in a permanent area is called "Persistence." This permanent data is called as Persistent data.

- * As, a Java Programmer, if we want to store the data permanently then we can make use of a file as a storage area.
- * Java programmer uses I/O streams for reading & writing the data using a file. we call the data format as a "Text."
- * A Java Programmer can also store an object into a file. Later, when it is required we can read the object back from a file. This process of reading and writing objects is called "Serialization".

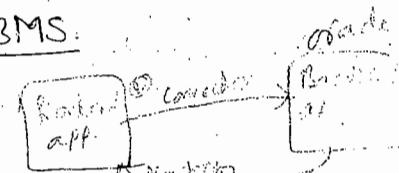


working with files is having the following drawbacks:

1. files are complex, w. more overhead. programmer has to care about file opening & file closing.
2. files doesn't support query language. So, Reading & writing the data using file is complex.
3. It is not possible to maintain the Relationship b/w the data stored in files.

→ In order to overcome the above drawbacks, we are shifted from FMS into DBMS.

why JDBC?



1. when a front-end application connecting with back-end, at initial days, with front-end application used set of functions given by database vendor to connect with the database.
2. Even today, A C & C++ applications uses "orcl.h" header file given by "oracle" to connect with the oracle database.
3. The drawback of using vendor given functions (API) is, the front end application becomes Database Dependent.
4. In order to overcome the above drawback, with professionals from Top MNC Companies, a group was formed called "X/Open" community.
5. The X/Open group was formed, to make a front-end application as database Independent.
6. X/Open group suggested a set of functions called as X/Open API. But, this API was

selected by the hardware vendors because, the implementation requires ALP (Assembly Language programming). So, X/Open group was failed.

7. After X/Open failed, Microsoft itself has formed another group with its own Professions was ODBC group (Open DataBase Connectivity)
 8. ODBC group created a set of functions called as ODBC API and the DataBase vendors accepted ODBC API. Because, the Implementation of ODBC requires 'C' language with pointers coding. It means ODBC API was success.
 9. If a front-end application uses ODBC then it can connect with any database.
 10. Java application uses ODBC. Then we have following two drawbacks:
 - (i) Java has no pointers but, ODBC is in pointers. So, non-pointers to pointers conversion is provi required. It becomes slow connectivity from Java to DataBase.
 - (ii) Java is platform Independent but, ODBC is platform Dependent. So, if Java uses ODBC then Java also becomes platform Dependent. This is against Java principle i.e., write once Run Anywhere.
- * In order to overcome the above drawbacks, SUN micro systems introduced its own technology as "JDBC"

as front-end applications in Real-time?

Ans:- If C & C++ are used as front-end then the front-end application becomes Database Dependent. Because, of this reason both C & C++ are not suitable as front-end for real-time.

* what is the difference b/w ODBC & JDBC?

Ans:- 1. ODBC is platform dependent but JDBC is platform independent.

2. ODBC implementation is in 'C' pointers, But JDBC implementation is in Java language.

→ what is JDBC?

1. JDBC is not a separate programming language.

2. JDBC is not an operating system.

3. JDBC is an in-built technology of Java SE.

4. we can work with JDBC technology by installing JDK S/W.

14/6/11 Definition:-

JDBC = API + SQL Driver

JDBC is an API specification, used for transferring SQL Commands between Java and Database.

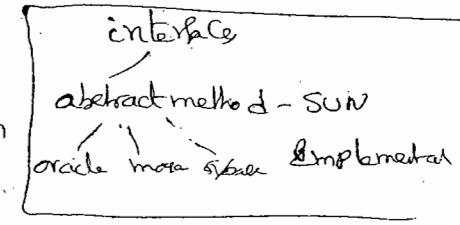
→ JDBC acts as a vehicle to transfer the data "to and fro" between a Java application & a Database.

→ "SUN" Microsystems given JDBC technology for the following two kinds of people.

1. Application writers — JDBC API

2. Driver writers — SQL Driver

- Application writers are responsible for developing applications (programs). For this kind of people, "SUN" microsystems has given JDBC API
- Driver writers are responsible to develop JDBC drivers. For this kind of people, "SUN" microsystems has given JDBC specification.
- JDBC API contains two Packages:
 - 1. `java.sql` ↗ classes → & its implementation given by 'sun'
 ↗ interfaces → & its implemented by vendor
 - 2. `javax.sql`
- The above two packages contains classes & Interfaces for developing JDBC applications.
- JDBC specification contains Rules & guidelines & these rules are followed by the Driver Vendors. (oracle, IBM, mysql...)
- JDBC Driver Vendor provides implementations for the interfaces given in the JDBC API.
- Driver vendors should follow the specification given by SUN, while creating JDBC Driver.



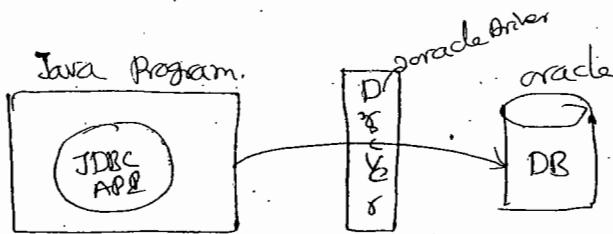
(Vendor) IBM → [DB2
 Sybase]
 ↗ It follows rules & guidelines by "sun"

~~Q~~ why 'SUN' microsystems has not given implementations for interfaces in JDBC?

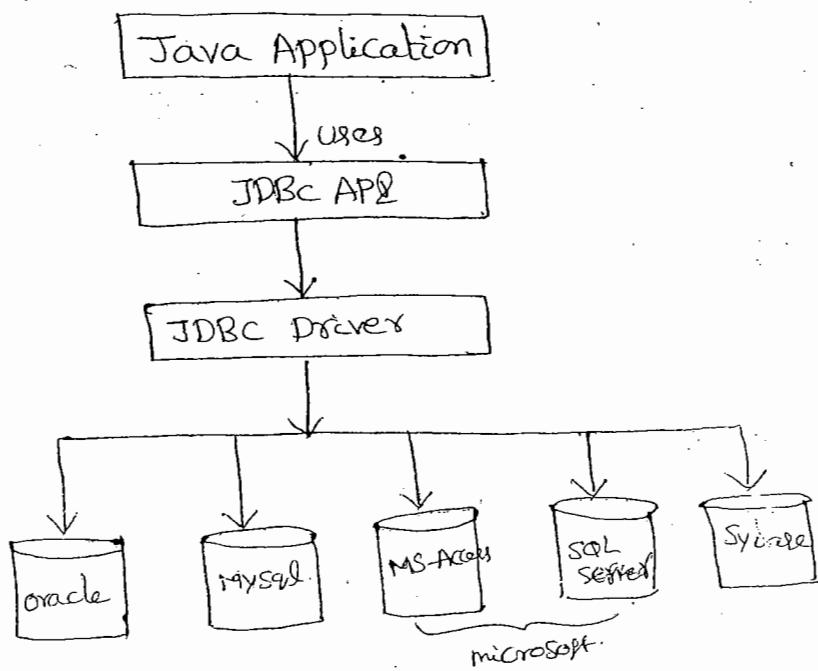
Ans:- If 'SUN' microsystem provided implementation also then, the Java program always connects with a single Database. It makes a Java program as Database Dependent. Because, of this reason 'Sun' has not provided implementations.
 * If vendors are providing implementations then Java

* what is JDBC

Ans: A JDBC Driver is a S/w Program, created by Vendors and it acts as a mediator for connecting Java and Database.



- * While creating JDBC Driver programs, vendors can use either 100% Java Coding (or) partially Java and partially some other native language.
- * we have both partial JDBC drivers and pure JDBC drivers.
- * A JDBC Driver is mandatory, while connecting from Java to a Database.



Note:- whenever a Java programmer wants to connect with a database then, Java programmer need the following two components.
1. JDBC API
2. JDBC Driver

Types of JDBC Drivers.

There are 4 types of Drivers.

1. JDBC-ODBC Bridge Driver.
 2. Native-API Partly Java Driver
 3. Net protocol Pure Java Driver
 4. Native protocol Pure Java Driver.
- } Partial Java Drivers
} pure Java Drivers.

Common points :-

1. Type-I and Type-II are partial Java Drivers. Type-II & Type-IV are pure Java Drivers. because Java has no pointers & ODBC has C+Pointers
2. Type-I is the slowest Driver, and Type-IV is the fastest Driver among the 4 types of drivers.
3. Type-I, II & IV follows 2-tier architecture model. But, Type-III follows 3-tier model: (Java | ^{middle} server | DB)

classes and Interfaces java.sql package :-

Interfaces:

- ✓ 1. Driver
- ✓ 2. Connection
- ✓ 3. Statement
- 4. PreparedStatement
- 5. CallableStatement
- ✓ 6. ResultSet
- 7. ResultSetMetaData
- 8. DatabaseMetaData
- 9. ParameterMetaData.

classes:-

- ✓ 1. DriverManager
- 2. DriverPropertyInfo

4. Time

5. TimeStamp

6. Types

package contains classes,
interfaces, exceptions & sub-packages

Exceptions :-

1. SQLException
2. SQLWarning
3. DataTruncation
4. BatchUpdateException

15/6/11

→ TYPE-1 :-

SUN

↓
interfaces

Vendors

↓
implementation

for real-time.

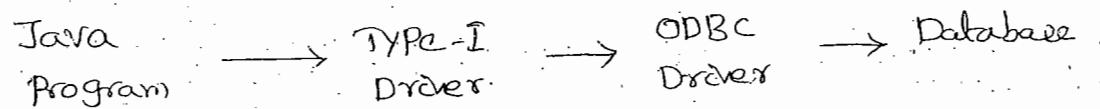
only for testing → i.e., Java is connected with db (or) not.
 not for real-time

TYPE-1 : JDBC-ODBC Bridge Driver :-

1. This type of driver is given by "SUN" micro Systems.
2. "SUN" given this driver for only "Testing" whether a Java application is connecting with a database (or) not.
3. This Type-1 driver is not suitable for production environment (Real-time).
4. The functionality of this driver is, it converts JDBC calls (functions) into ODBC calls (functions), while connecting with a DataBase.

5. JDBC calls are implemented in Java Programming and ODBC calls are implemented in C language with Pointers. So, the conversion takes some amount of time. So, this Type-I driver is slow.

6. Type-I driver internally uses ODBC driver for connecting with a database.

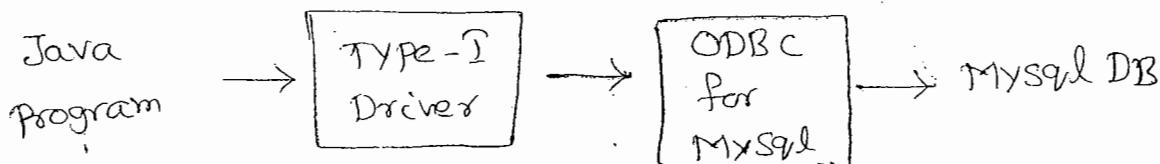
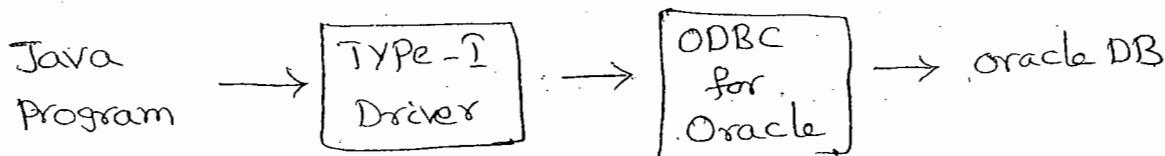


* In the middle Type-I Driver duties, converting non-pointers code into pointers code and in reverse pointers code into non-pointers.

* ODBC driver is a database dependent driver. It means we have separate ODBC driver for each database.

* The relationship between ODBC driver and a database is 1:1 i.e., one ODBC driver has one db.

* Type-I driver is common for all databases. It means the relationship b/w Type-I driver and ODBC driver is 1:n.



Advantages :- (Pros)

1. This Type-I driver is database independent.
2. This Type-I driver automatically comes along with JDK &/w.

These are :-

Disadvantages:- (cons)

1. This Type-I driver is the slowest driver among all JDBC drivers.
2. This Type-I driver internally uses ODBC, so this driver is platform-dependent.
3. This Type-I driver is not suitable for Real-time applications.

Note:-

* The Type-I driver is suitable only for local system applications.

* SUN Microsystems has given the following driver class.

Sun.jdbc.odbc.JdbcOdbcDriver → Fully Qualified class name
 |
 |
 Packagename classname (FQCN)

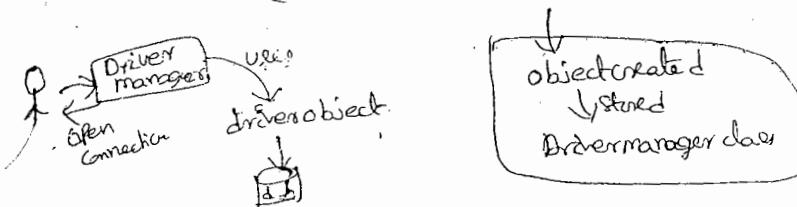
Steps to write a JDBC program:-

Step-1:- Import JDBC packages (API)

Syn:-
import java.sql.*;
import javax.sql.*;

Step-2:- Load and Register a JDBC driver

Syn:- 1. Class.forName ("Sun.jdbc.odbc.JdbcOdbcDriver");



2. DriverManager.registerDriver (Object of driver class)

For ex:-

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

(or)

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

```
DriverManager.registerDriver(od);
```

Step-3:-

Obtain a connection with the database.

Syn:- Connection con = DriverManager.getConnection(url, username, password);

Step-4:-

→ Create a statement object.

→ This statement object has the methods to transfer SQL Commands from Java to database.

Syn:- Statement stmt = con.createStatement();

Step-5:-

Execute the SQL commands on Database

Syn:- output = stmt.execute("sql command");

Step-6:-

Print the output.

Syn:- System.out.println(output);

Step-7:-

close the connection with Database.

Syn:- con.close();

1. executeUpdate ("sql command");
2. executeQuery ("sql command");
3. execute ("sql command");

1. executeUpdate() :-

- * This method is used for executing non-selecting options on db.
- * Non-select options means both DDL & DML options on db.
- * This method returns an integer values that integer value can be either -1 or a positive number (> 0)
- * This method returns -1, if the command executed on db is a DDL command. If the command is DML then this method returns a positive integer.

Syntax:-

```
int x = stmt.executeUpdate("sql command");
```

For ex:-

```
1. int x = stmt.executeUpdate("create table sathya (sid number(5), sname varchar(10));");
```

→ Here, if the table is successfully created on database then the variable 'x' contain -1 i.e., $x = -1$ (DDL cmd).

```
2. int x = stmt.executeUpdate("insert into sathya values(111, 'abcd')");
```

Here if the row is successfully inserted into the table then the 'x' contains 1 i.e., $x = 1$. (DML cmd)

2. executeQuery() :-

- * This method is only used for executing select option on database.

- * In JDBC, the selected data or records from the table will be stored in a ResultSetObject. It means the return type of this class is ResultSet.

Syntax

ResultSet rs = stmt.executeQuery("sql command");

Ex:-

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

3. execute()

* This method is used for both executing Select & non-Select options on database.

* This method returns boolean. If select option is executed then this method returns "True". If non-select option command is executed then this method return "False".

Syntax

boolean b = stmt.execute("sql command");

Ex:-

1. boolean b = stmt.execute("alter table ---");

Here b contains False.

2. boolean b = stmt.execute("delete from ---");

Here b contains False.

3. boolean b = stmt.execute("select * from customer");

Here b contains True.

→ The following JDBC program is for creating a table in oracle database, using type1 driver.

// Step - 1

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

```
class JdbcExample
```

```
{
```

```
p.s.v.m( String [] args) throws Exception
```

```
{
```

// Step - 2

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

// Step - 3

```
DriverManager.getConnection(
```

//Step-4

Statement stmt = con.createStatement();

//Step-5

int x = stmt.executeUpdate("create table test123

(Sno number(3), Sname varchar(10), marks number(3));

//it returns -1.

//Step-6.

if (x == -1)

S.O.P ("table created");

//Step-7

stmt.close();

con.close();

S.O.P ("connection closed");

}

→ JDBC handle with checked Exceptions.

O/p :-

D:\> javac JdbcExample1.java

D:\> java JdbcExample1 ↴

Connection created.

Table created

Connection closed.

SQL > desc test123;

Name

SNO

SName

Marks

D:\> java JdbcExample1 ↴

Name already used by an object, exception

occurred.

→ If we want to delete the table go to SQL

SQL > drop table test123 ↵

Table is dropped.

7) 6th DSN (Data Source Name) :-

Q what is DSN?

Ans.: A DSN is a datastructure, it contains information of a db's and it is used by ODBC driver, for opening connection with the database.

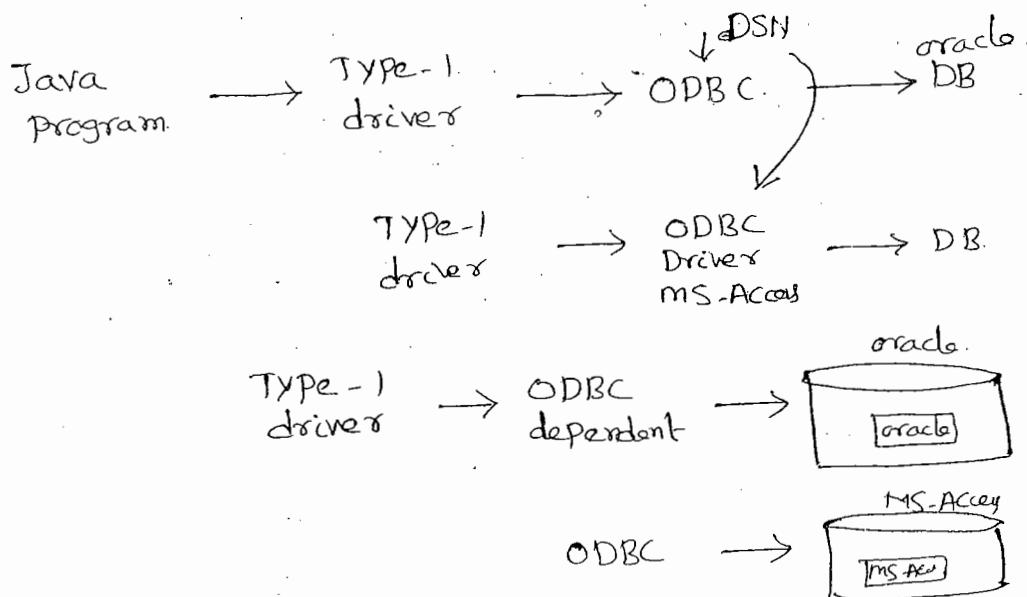
→ without DSN, an ODBC driver cannot open connection with a database.

→ DSN's are of 3 types

1. System DSN

User DSN

3. File DSN.



- * A System DSN is a global DSN, it is sharable for all users of the system.

- * An user DSN is a local DSN. & it is only for one user of the system.

this type of DSN then the DSN information will be stored in file & we can transfer this file from one system to another system in a n/w.

- * In file DSN, the DSN will be stored in a file & its extension is .DSN.

Steps to create DSN:

Start → settings ↓

Control panel



Administrative Tools



Data Sources (ODBC)



Add Button



Select Microsoft ODBC for oracle



Finish



Select Data Source Name:



any db we can take.

Username:

↓
OK.

Note:-

1. while creating DSN, if any error like oracle client & n/w components not found is occurred then we need to verify whether path variable contains oracle path or not, in the environment variables.
2. Even though, if oracle path exists and if get the

error then we need to copy or move the oracle Path to the 1st place in that value.

→ The following Jdbc program is for inserting dynamic values into a table using Type-1 driver.

// step1

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

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

```
class JdbcExample2
```

```
{
```

```
    p.s.v.m (String [ ] args) throws Exception
```

```
{
```

// step-2

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

// step-3

```
    Connection con = DriverManager.getConnection(
```

```
        "sun.jdbc.odbc:oradsn", "scott", "tiger")
```

```
    S.O.P ("connection created");
```

// step-4

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

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

```
    S.O.P ("enter student no");
```

```
    int n = s.nextInt();
```

```
    S.O.P ("enter student name");
```

```
    String str = s.next();
```

```
    S.O.P ("Enter marks");
```

```
    int m = s.nextInt();
```

// step-5

```
    int x = stmt.executeUpdate("insert into test123  
        values ('" + n + "', '" + str + "', " + m + ")");
```

// step-6

```
    S.O.P (" " + x + " row inserted");
```

```
        stmt.close();
        con.close();
    }
}
```

9/P:-

```
javac jdbcExample.java &
```

```
java jdbcExample2 <
```

Connection created

Enter student no:

303

Enter student name

ABC

Enter student marks

200

1 row is inserted.

In SQL plus

```
Select * from test123;
```

Sno	sname	smarks
303	ABC	200

Selecting Data from Table:-

* we need to store the data selected from a table
onto ResultSet Object.

* whenever data is stored in a ResultSet object
then the ResultSet cursor points at before 1st record
by default.

- * To move the cursor to the next records one by one then we need to call `next()` method.
- * `next()` method returns either true or false. It returns true, if next record is available and otherwise it returns false.
- * If you want to read the data from each row after `ResultSet` then we need to call `getx()` methods.

For ex:-

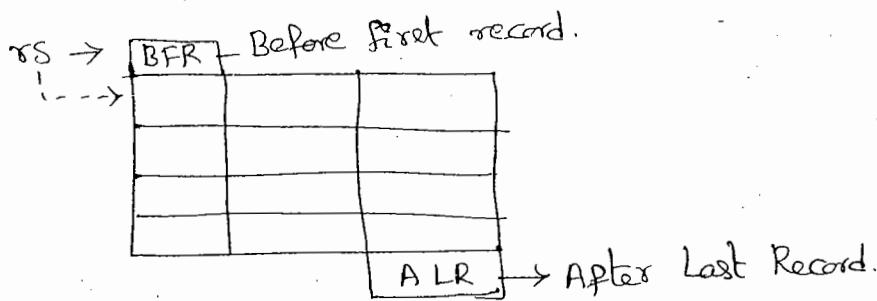
`getInt()` for int value

`getString()` for string value etc.

- * If we do not know the value type then we use common method as `getString()`.

Ex:-

`ResultSet rs = stmt.executeQuery("select * from test123");`



`while(rs.next())`

{

`S.O.P(rs.getInt(1) + " " + rs.getString(2) + "`
`+ rs.getInt(3));`

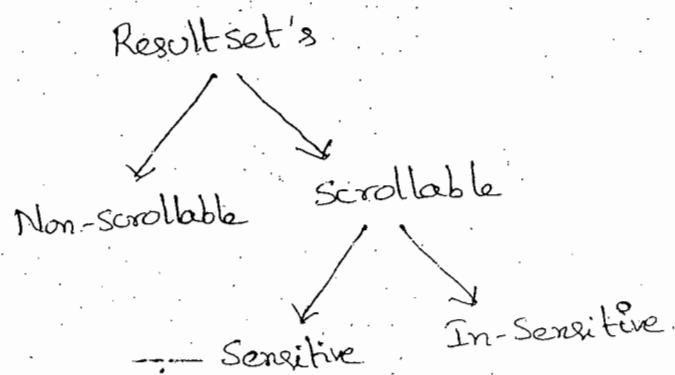
}

18/61"

* In JDBC, Result Sets are of two types:

1. Non-Scrollable ResultSet (default)
2. Scrollable ResultSet

* By default, every Resultset is a Non-Scrollable. But, it is possible to convert from Non-Scrollable into scrollable type.



→ The following JDBC Program is for selecting the data from table using Type-I driver.

```
// JDBC Example3.java
```

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

```
class JDBCExample3
```

```
{
```

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

```
{
```

```
        Sun.jdbc.odbc.JdbcOdbcDriver od = new
```

```
        Sun.jdbc.odbc.JdbcOdbcDriver();
```

```
        DriverManager.registerDriver(od);
```

```
        Connection con = DriverManager.getConnection("
```

```
            jdbc:odbc:oradsn", "scott", "tiger")
```

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

```

    ResultSet rs = stmt.executeQuery("select * from test123");
    while(rs.next())
    {
        System.out.println(rs.getInt(1) + " " + rs.getString(2)
                           + " " + rs.getInt(3));
        System.out.println("-----");
    }
    rs.close();
    stmt.close();
    con.close();
}
//main
}
//class

```

O/P:-

javac JdbcExample3.java

java JdbcExample3

101	abc	345
202	Sathya	780
303	tech	560

Explanation of execute() method :-

- * execute() method is used for both select & non-select operations on database.
- * execute() method is not actually executing the command. Instead, it verifies the command.
- * If the command is "Select" then returns True and if the command is "non-select" then returns False.
- * execute() method functionality is called "Lazy Execution".

* If we use ~~new~~ new another method also for executing the sql command on db.

* For, non-select, operations, the method is

getUpdateCount()

* For, Select operations, the method is getResultSet().

For example:-

Ex:-

```
boolean b = stmt.execute("select * from emp");
```

```
if (b == true)
```

```
{
```

```
ResultSet rs = stmt.getResultSet();
```

```
while (rs.next())
```

```
{
```

```
    //
```

```
    //
```

```
}
```

```
3.
```

Ex:-

```
boolean b = stmt.execute(" delete from emp where eno='101'");
```

(or)

Create

(or)

insert

```
if (b == false)
```

```
{
```

```
int x = stmt.getUpdateCount();
```

```
if (x == -1)
```

DDL :- -1

DML :- Positive number.

```
S.O.P ("DDL cmd executed");
```

else

```
S.O.P ("DML cmd executed");
```

```
3
```

→ The following JDBC program is for executing the given command on database.

// JdbcExample4.java

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

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

```
class JdbcExample4
```

```
{
```

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

```
{
```

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

```
    Connection con = DriverManager.getConnection ("jdbc:odt  
oradsn", "scott", "tiger");
```

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

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

```
    System.out.println ("Enter a command of your choice");
```

```
    String str = s.nextLine();
```

→ It is for reading the command
in next line.

```
    boolean b = stmt.execute (str);
```

```
    if (b == true)
```

```
{
```

```
    ResultSet rs = stmt.getResultSet ();
```

```
    while (rs.next ())
```

```
{
```

```
        System.out.println (rs.getInt (1) + " " + rs.getString  
        " " + rs.getInt (3));
```

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

```
} //while
```

```
    rs.close ();
```

```
} //if
```

```

use
{
    boolean
    int x = stmt.executeUpdate();
    if (x == -1)
        S.O.P ("DDL cmd executed");
    else
        S.O.P ("DML cmd executed");
}
//else
stmt.close();
con.close();
} //main
} //class.

```

O/P:

javac JdbcExample4.java

javac JdbcExample4 ↵

Enter a command of your choice

insert into test123 values (404, 'tech', 670)

DML cmd executed.

→ For verifying go to SQLplus

Select * from test123.

no	name	marks
404	tech	670

D:\> java JdbcExample4

enter a command of your choice.

Select * from test123

101	abc	345
202	sathya	780
303	tech.	560
404	tech	670

→ Enter a command of your choice.

create table Java (SID number(3), Sname varchar2(10))

DDL cmd executed

Go to SQLplus.

desc java;

SID

Sname

Type-2 :- Native-API Partly Java driver

This driver translates JDBC calls into direct database calls for connecting with the db.

* This driver is Database Dependent. It means their ratio b/w Driver and db is 1:1.

* Type-2 driver doesn't use any other driver in the middle. It directly connects with the db.

Java JDBC calls → Type-2 Native calls → Database
Program Driver

Java program → Type-2 Driver for Oracle → Oracle

Java program → Type-2 Driver for MySQL → MySQL

Advantage :-

1. This driver is faster than Type-1 driver.

Disadvantages :-

1. This Type-2 driver is Database Dependent & also platform dependent.

3. This driver is slow compare with Type-3 & Type-4 drivers.
 4. This Type-2 driver is not a portable driver.

20|6|11

Type - 2 for oracle :-

Oracle database vendors has provided two JDBC drivers for connecting from Java to Oracle db.

1. oracle oci driver
 2. oracle thin driver

* Occ driver belongs to type-2 category & then driver belongs to type-4 driver.

- * oracle oci driver converts jdbc calls into oci calls (Oracle Call Interface), for establishing the communication b/w Java & db.

Java program → JDBC calls → oracleoci driver. → OCI calls → oracle DB.

→ The following information is required to connect from java to oracle db using type-2 driver.

Driver class name : oracle.jdbc.OracleDriver

url : jdbc:oracle:oci:@<SID>

↓
Service ID.

Username : scott

password : Tiger

→ The follo

→ whenever oracle database installed then along with physical db, one logical db also created.

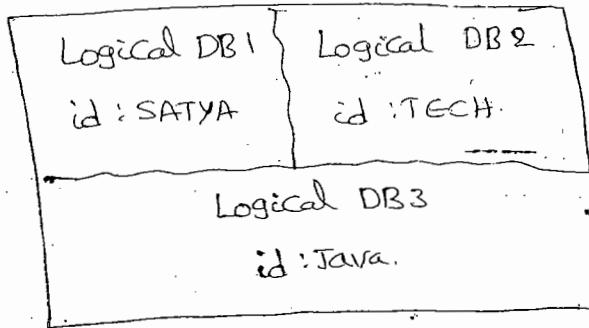
→ when we are working with oracle, it means that we are working with logical db but not physical db directly.

→ for each logical db, there is a unique ID to identify it.

→ this ID is called Service ID (SID).

→ By default when database is installed one logical db is created & later any no. of logical database can be created.

physical DB of oracle



Different ways of finding SID:

1. At SQL plus type the following command to know SID.

SQL> select * from global_name;

GLOBAL NAME

SATYA.OS.ORACLE.com.

2. start → settings → control-panel

↓
Administrative tools

↓
Services

↓
Select oracle service from the lists find
sid (OracleService SATYA)
c.D.

in the file.

1 admin folder * find service name

→ type-2 driver is a platform dependent because it uses different databases.

Java program → type-2 driver → SQL calls for Sybase

→ The following example is for deleting an employee from emp table using type-2 driver.

JdbcExample5.java

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

```
class JdbcExample5
```

```
{
```

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

```
    {
        Class.forName("oracle.jdbc.oracleDriver");
```

```
        Connection con = DriverManager.getConnection
```

```
( "jdbc:oracle:oci:@SATYA",
    "scott", "tiger");
```

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

```
        int x = stmt.executeUpdate("delete from
            emp where empno = 7876");
```

```
        System.out.println(x + " row deleted");
```

```
        stmt.close();
```

```
        con.close();
```

```
}
```

D/P/C:1> javac JdbcExamples.java

java JdbcExamples

Exception in thread main java.lang.ClassNotFoundException

→ If we run the above Java program then we will get ClassNotFoundException. Because we are using oracleDriver class, it is not given Java API. So, JVM unable to recognize the class. Because of this reason an exception is thrown.

→ In order to inform JVM about a third party class (Not Part of API). we need to set classpath.

→ we need to set ojdbc14.jar given by Oracle into classpath. So that JVM goes to that jar file & takes its bytecode (class code) & loads it into JVM.

→ C:1> set classpath=D:\oracle\ora92\jdbc\lib\ojdbc.jar;

↓
-1.classpath

D:1>java JdbcExamples

1 row deleted

do not modify
previous value of the
classpath.

Q When path & classpath settings are required?

Ans: While, working with any .exe files (or) a batch file (or) command file, if an OS is not recognizing them we will get "not recognized" error. To solve this error, we use path setting.

Ex: javac test.java
"javac is not recognized"
While compiling a Java program, if we get "package does not exist" error (or) while executing a

Java program, if we get no class definition found error (or) if we get ClassNotFoundException then we need to set classpath.

as related to JVM.

Q When ClassNotFoundException is raised by the JVM?

Ans: Two reasons.

1. Loading a class given by Java API, but the class is not included in that package then we will get ClassNotFoundException.

2. For example,

```
class.forName("java.applet.String");
```

2. In a Java program, if we use a class which is not a part of Java API then ClassNotFoundException is raised.

```
import oracle.jdbc.*;
```

```
class Test
```

```
{
```

```
p.s.v.m(String[] args)
```

```
{
```

```
}
```

```
javac Test.java.
```

i: Package oracle.jdbc does not exist.

c: i) Set classpath = d:\ - - - - -

Note:-

c: i) Path = d:\Program Files\java\jdk1.8\bin // Path

It is used for not overwritten by previous path.

→ c: i) set classpath = %classpath% ; d:\

c: i) set Path = %path%; d:\

→ C:\> java Test

Exception in thread "main": java.lang.NoClassDefFoundError
This problem is find when we are setting the classpath in
the current directory

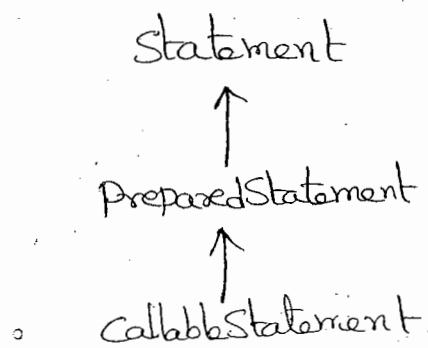
→ C:\> Set classpath = ?classpath? ;
↳ current location setting
classpath

In this we set classpath in the current directory.

Note:- For SDK tools, only, we set the path. For other
exceptions, we set the classpath.

Prepared Statement Interface :-

1. prepared statement is an interface given in java.sql package and it is extended from Statement
2. In JDBC, there are 3 types of statements:
 1. Statement
 2. PreparedStatement
 3. CallableStatement



~~Q~~ Differences b/w Statement & PreparedStatement:

Ans:-

Statement

1. Statement interface doesn't support binary datatypes of the database. So, we

PreparedStatement

1. PreparedStatement supports binary types of the database. So, we can insert a picture into a database.

2. In Statement interface, if we want to set dynamic values into SQL query, then very complex command is required.

3. Statement interface compiles the SQL command, for each time before executing internally. Even though, same command is executing repeatedly, but Stmt interface compiles the command internally each time. It decreases the performance at other side. (or reduces)

2. In PreparedStatement, a simple query is required for setting dynamic values into the query. because, in PreparedStatement, we can use "?" symbol in the SQL Command.

3. In PreparedStatement, it compiles the SQL command for only once. Later, it can be executed for any no. of times, by without recompiling once again. It increases the performance. It means PreparedStatement is faster than Statement interface.

PreparedStatement:-

Syntax :-

PreparedStatement pstmt = con.prepareStatement ("SQL Command");

for ex:-

PreparedStatement pstmt = con.prepareStatement ("insert into test23
values (?, ?, ?)");

'?' - place Resolution Operator (or) Replacing operator

It is also called as "pre-compiling"

Ex:- PreparedStatement pstmt = con.prepareStatement ("insert into
test23 values (?, ?, ?)");

```
for (i=1; i<=10; i++)  
{  
    pstmt.setInt (1, i);  
    pstmt.setString (2, str);  
}
```

```
psmt = stmt.executeQuery();
int i = psmt.executeUpdate();
```

Q) //

→ while creating PreparedStatement object we are passing the SQL query with a special symbol called '?'. we call this '?' symbol as place resolution operator (or) replacement operator.

* At run time, if we want to set values into '?' place, we need to setXXX() methods given by PreparedStatement

→ The following JDBC program is for updating salary of an employee by taking input values at Run-time, Using type-2 driver.

```
// JdbcExample7.java
```

```
import java.sql.*;
import java.util.*;
class JdbcExample7
{
    public void main(String[] args) throws Exception
    {
        Class.forName("oracle.jdbc.OracleDriver");
        Connection con = DriverManager.getConnection("jdbc:oracle:oci:@Satya", "scott", "tiger");
        PreparedStatement psmt = con.prepareStatement(
            "update emp set sal=? where empno=?");
    }
}
```

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

```
s.o.p("Enter salary");
```

```
double d = s.nextDouble();
```

```
s.o.p("Enter empno");
```

```
int n = s.nextInt();
```

```

    pstmt.setDouble(1, d);
    pstmt.setInt(2, eno);
}
int x = pstmt.executeUpdate();
System.out.println(" " + x + " row updated");
pstmt.close();
con.close();
}
}
}

```

O/P:-

```

D:\> javac JdbcExample7.java
D:\> java JdbcExample7
Exception in thread "main" java.lang.ClassNotFoundException:
        at java.net.URLClassLoader$1.run(URLClassLoader.java:202)
        at java.net.URLClassLoader$1.run(URLClassLoader.java:190)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:188)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:307)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:307)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:266)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:275)
        at JdbcExample7.main(JdbcExample7.java:14)

```

D:\> java JdbcExample7

Enter emp salary

9999

Enter eno

7788

1 row updated

Verify SQLplus

select sal from emp where empno = 7788;

SAL

9999

→ Inserting a picture:-

1. Inserting a picture means storing binary data of a picture into a column of a table.

2. If we want to store binary data into a column then, that column datatype must be BLOB (Binary large object)

3. Directly we can't insert a picture into a table
is only supported by PreparedStatement, but not possible with
Statement.

4. To insert binary data of a picture, into a database
column then we need to call a method setBinaryStream()
given by PreparedStatement interface.
5. For inserting a picture into a table, we need the
following two class objects.

1. File class
2. FileInputStream class

6. The above two classes are given in `java.io` package.

Example:

`PreparedStatement pstmt = psdb.prepareStatement ("insert into companies
values (?, ?)");`

```
pstmt.setString(1, "abc Ltd");
File f = new File("c:/duke.gif"); // TO find size of
fileInputStream fis = new FileInputStream(f);
int l = (int)f.length(); // l must be in int that's
why we do explicit casting
pstmt.setBinaryStream(2, fis, l); // int
int x = pstmt.executeUpdate();
System.out.println(x + " row inserted");
```

→ Image size should be less than < 4 KB.

→ The following JDBC program is for inserting a
Company name and its logo into a table called Companies
using PreparedStatement.

The following command is required for
store creating a table in oracle database

SQL > Create Table Companies (company_name VARCHAR(10),
logo BLOB);

Table created.

SQL >

//PictureInsert.java

import java.io.*;

import java.sql.*;

class PictureInsert

{

public void main(String[] args) throws Exception

{

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

Connection con = DriverManager.getConnection(

"jdbc:oracle:oci:@SATYA", "scott", "Tiger");

PreparedStatement pst = con.prepareStatement(

"insert into companies values(?,?)");

pst.setString(1, "abc Ltd.");

//creating an object for file

File f = new File("c:/duke.gif");

//finding size of the file.

long l = f.length();

int s = (int) l; //explicit type cast.

FileInputStream fis = new FileInputStream(f);

//creating file object reading the file

//setting binary data into 2nd question mark.

pst.setBinaryStream(2, fis, s);

int i = pst.executeUpdate();

S.o.p(i + " row inserted");

fis.close();

```
    pst.close();
    con.close();
}
```

O/P:-

```
javac pictureInsert.java
```

```
java pictureInsert
```

1 row inserted

```
SQL> Select * from companies;
```

column or attribute type cannot be displayed by SQL Plus

```
SQL> Select companyname from companies;
```

COMPANYNAME

abc ltd.

Q3) 6/11

→ Selecting a Picture:-

1. In order to select a picture from a database table, we need to use PreparedStatement.

2. Selecting a picture means reading binary data from a

table & constructing an image by using that binary data.

3. for Constructing an image, we need FileOutputStream.

4. for Constructing an image, we need to writing binary data into a file.

5. whenever, binary data is selected from a table then,

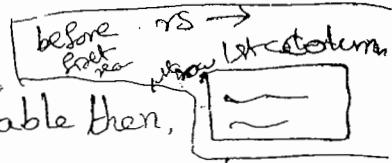
ResultSet stores the binary data into a Column of

ResultSet. After that, if we want to get the binary

data from ResultSet then, we need to use getBinaryStream()

method.

6. while, creating (or) Constructing a new image file with binary. If then we need to check whether the binary data



→ The following JDBC program is for selecting a picture from the table using TYPE-2 driver.

// PictureSelect.java

```
import java.io.*;
import java.sql.*;
class PictureSelect
{
    public static void main (String [] args) throws Exception
    {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con= DriverManager.getConnection("jdbc:oracle:
Oci:@SATYA", "scott", "tiger");
        PreparedStatement pst = con.prepareStatement ("select
Logo from Companies where Comprname=?");
        pst.setString(1, "abc ltd");
        ResultSet rs = pst.executeQuery();
        rs.next();
        InputStream is = rs.getBinaryStream(1);
        FileOutputStream fos = new FileOutputStream(
"E:/test/sample.gif");
        int k;
        while ((k = is.read ()) != -1)
        {
            fos.write(k);
        }
        System.out.println ("Image received");
    }
}
```

```
fos.close();
pst.close();
con.close();
```

{
}O/P :-

```
javac PictureSelect.java
```

```
java PictureSelect
```

Image received.

for checking image:-

+ → Test folder

+ → Sample.gif

↓
open.

duke.gif, Sample.gif are same.



P implements Satya

{

}

Satya s = new Test();

→ classes are created by vendors. Interfaces are created by "sun"

~~Q~~ In Connection con = DriverManager.getConnection(); statement,

'con' is a reference (or) an object.

Ans: 'con' is not an object. It is a reference.

* Internally, whenever getConnection() method is called,

then it internally calls connect() method of Driver.

The connect() method returns an object of connection class (oracle.Connection class object).

returned object into a reference of Connection interface.

Finally, that reference is given to Java Program.

Connection con = DriverManager.getConnection(---)

class DriverManager

{

public static Connection getConnection(---)

{

Connection con = calls connect() of driver class.

(4)

returns con;

}

(3)

class oracleDriver implements Driver

{

connect()

{

→ Create OracleConnection

class object

→ returns OracleConnection

object

}

this process called Call-back mechanism

Q what is JDBC call back mechanism?

A It means calling a method, creating an object internally for a class and returning an interface reference back to the Java Program.

Ex, when we call getConnection() of DriverManager, then, internally OracleConnection object is created (^{for Oracle}) and to the java program Connection reference is given back. It is a "call-back mechanism".

24) 6/11

Type-4 Native-protocol Pure Java Driver :-

- * This driver is a pure Java driver. It means this driver implementation is completely done using Java Port.
 - * This Type-4 driver doesn't convert Jdbc calls into any other type of calls.
 - * This type-4 driver uses the following 3 informations, for connecting a Java application with database.
 1. protocol of the database Server
 2. IP address of the database System
 3. Port number of the database Server
 - * This Type-4 driver doesn't use any mediator driver for connecting from Java to a database.

Java program JDBC API → TYPE - 4 Native protocol Database drivers

- * For example, while connecting from Java to oracle db, OracleDriver uses ttc (two-task-common) protocol for connecting with oracle Database.

- * for each database, we have a standard (or) separate protocol and port number. So, these Type-4 drivers are database dependent drivers.
 - * For example, Oracle Database Server runs on a standard port no. called 1521. MySQL Database Server runs on port no. 3306.

1. Type-4 driver is the fastest driver among all types of Jdbc drivers.
2. Type-4 driver is a pure-Java driver so, it is platform-independent.
3. Type-4 driver is suitable for internet applications.
It means ~~this~~ driver is used in production environment (Real-time).

Disadvantage:-

1. This Type-4 driver is a database dependent driver.

Type-4 driver for Oracle :-

* Oracle database vendor has provided two Jdbc drivers for connecting with oracle.

1. Oracle oci driver
2. Oracle thin driver.

* Oci driver is a type-2 driver and converts Jdbc calls into native calls.

* Thin driver is a type-4 driver and it directly connects with the database for any where in the n/w by using protocol, IP address & port no.

* Between oci driver & thin driver, there is only a change in the url. Remaining Driver class, username & password are one and same.

* Type-2 OCI driver URL is:

jdbc:oracle:oci:@satya
<sid>

* Thin driver URL is:

jdbc:oracle:thin:@<IP address>:<port no>:<sid>

oracle: jdbc:oracle:thin:@spaddress:1521:satya

MySQL: jdbc:mysql://localhost:3306/beeh

→ The following JDBC program is for selecting data from a table using Type-4 Oracle driver.

//JdbcExample8.java

```
import java.sql.*;
class JdbcExample8
{
    public static void main(String[] args) throws Exception
    {
        Driver d = new oracle.jdbc.OracleDriver();
        DriverManager.registerDriver(d);
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:satya", "scott", "tiger");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("Select * from dept");
        while(rs.next())
        {
            System.out.println(rs.getString(1) + " " + rs.getString(2) +
                               " " + rs.getString(3));
        }
    }
}
```

```
    IS_CLOSEL  
    Stmt.close();  
    Con.close();
```

O/P:-

```
javac JdbcExample8.java
```

```
java JdbcExample8
```

10	ACCOUNTING	NEWYORK
11	SALES	DELHI
12		

Callable statement interface :-

1. callable statement is an interface given in java.sql package & it is extended from PreparedStatement interface.
2. In CallableStatement, there is one additional feature when compared with PreparedStatement.
3. The additional feature of CallableStatement is we can call both procedures & functions of a database.
4. CallableStatement is for both executing SQL commands and also for calling procedures & functions.

25/6/11
Different ways of creating Callable Statement :-

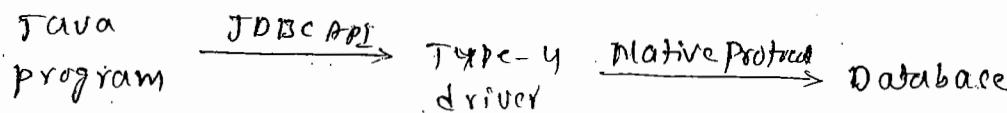
* CallableStatement is used for both executing SQL Commands on database and also for calling procedures and functions of the database.

24/6/11

Type-4

Native protocol pure Java driver

- This driver is a pure Java driver it means this driver implementation is completely done using Java code.
- This type-4 driver doesn't convert JDBC calls into any other type of calls.
- This type-4 driver uses the following three informations for connecting a Java application with a database.
 - i) protocol of the database server
 - ii) IP address of the database system
 - iii) port number of the database server
 - iv) This type-4 driver doesn't use any mediator driver for connecting from Java to a database.



⇒ For example,

- * While connecting from Java to Oracle database, Oracle Driver uses TTC protocol (two talk common) for connecting with oracle database. Here TTC is a native protocol of Oracle Database server.
- * For each database, we have a standard or separate protocol and port number so this type-4 drivers are database dependent drivers.

⇒ For example,

- * Oracle database server runs on a standard port number called 1521 and MySQL server runs on 3306 port number etc.

Advantages

- i) Type-4 driver is the fastest driver among all type of JDBC drivers.
- ii) Type-4 driver is a pure Java driver, so it is platform independent.
- iii) Type-4 driver is suitable for internet applications. It means these drivers used in production environment (real time).

Disadvantage

- This driver is a database dependent driver.

Type-4 Driver for Oracle

- Oracle database vendor has provided two JDBC drivers for connecting with Oracle
 - (i) Oracle OCI driver
 - (ii) Oracle thin Driver

oci driver is a type-2 driver and converts JDBC calls into native calls.

thin driver is a type-4 driver and it directly connects with the database from anywhere in the network by using protocol, IP address and port number.

Between oci driver and thin driver, there is only a change in the URL. Remaining driver class, username and password are same.

oci driver URL is : jdbc:oracle:oci:@sid

thin driver URL is : jdbc:oracle:thin:@(IP address):(port number)&sid

The following JDBC program is for selecting data from a table using type-4 Oracle driver ?

// JDBC Example.java

```
import java.sql.*;  
class JdbcExample  
{  
    public static void main (String args[]) throws Exception  
{  
        Driver d = new oracle.jdbc.OracleDriver();  
        DriverManager.registerDriver(d);  
        Connection con = DriverManager.getConnection ("jdbc:oracle:  
            thin:@localhost:1521:satya", "scott", "tiger");  
        Statement stmt = con.createStatement();  
        ResultSet rs = stmt.executeQuery ("select * from department");  
        while (rs.next ())  
        {  
            System.out.println (rs.getString (1) + " " + rs.getString (2) + " " + rs.getString (3));  
        }  
        rs.close ();  
        stmt.close ();  
        con.close ();  
    }  
}
```

CallableStatement Interface

- ⇒ CallableStatement is a interface given in java.sql; and it is extended from PreparedStatement interface.
- ⇒ In CallableStatement there is one additional feature when compared with PreparedStatement.
- ⇒ The additional feature of CallableStatement is, we can call both procedure and functions of a database.
- ⇒ CallableStatement is for both executing sql commands and also for calling procedures and functions.

Different ways of creating CallableStatement

- CallableStatement is used for both executing sql commands on database and also for calling procedure and functions of the database.
- ⇒ In order to get CallableStatement reference (object), we need to prepareCall() given by connection interface.
we have two syntaxes for creating CallableStatement object one for sql commands and other one is for calling procedures and functions.

Syntax 1 :-

```
CallableStatement cstmt = con.prepareStatement("sql command");
```

Syntax 2 :-

```
CallableStatement cstmt = con.prepareCall("{call proname  
           (params) / buname (params)}");
```

- ⇒ In programming languages, the difference between a procedure and function is, procedure does not return any value but a function returns a single value.

* According to database programming, the difference b/w a procedure and a function is, a procedure can return zero or more values. But a function exactly returns only one value.

Syntax to create a procedure

Create or replace procedure procedurerename (param1s any)
is

variable declarations;

Begin

stmts;

end;

parameters for a or a procedure are having three mode

① In

② Out ③ InOut

⇒ In parameters ^{can} be accepting the input, Out parameters are for producing the output and InOut parameters are both accepting input as well as producing output.

⇒ By default a parameter is ~~an~~ in mode

⇒ By depending no. of Out parameters we can say that a procedure is returning these many Out values

⇒ Inside the body of a procedure, we no need to write any return statement. Procedure implicitly returns the values.

Ex The following procedure in database accepts input as employee number and returns experience of that employee as output

DOL :- Dynamic User Allocation List

SQL (create or replace procedure 'experiencepro' (eno in number, e out number)

d1 date;
d2 date;

begin

select hiredate into d1 from emp where empno = eno;

Select sysdate into d2 from dual;

$$e := (d2 - d1) / 365;$$

end;

[we have to type above code in one file]

Creation

SQL @ Sample1

44 1

procedure created.

Note :- While creating procedures and functions in database,
Don't provide size for the parameters

7/6/10 Important points for calling a procedure or a function

→ While calling a procedure or a function from a Java application then we need to call setX()s and we need to pass value for in parameters.

→ If any Out parameters are there then we need to register the output parameters by calling register OutParameter() method.

→ To call a procedure or function of the database we need to use execute()

→ After executing the procedure or a function, we need to call getX() for reading out parameters

The following JDBC program is for calling the stored experience procedure of the database.

// CallableTest1.java

```
import java.sql.*;
import java.util.*;
class CallableTest1
{
    public static void main(String[] args) throws Exception
    {
        DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
        Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Sathyam", "scott", "tiger");
        CallableStatement cstmt = con.prepareCall("{call experience(?,?)}");
        Scanner s = new Scanner(System.in);
        System.out.println("enter emp number");
        int n = s.nextInt();
        cstmt.registerOutParameter(2, Types.INTEGER);
        cstmt.execute();
        int lc = (cstmt.getInt(2));
        System.out.println("output : experience of " + n + " is " + lc + " years");
        cstmt.close();
        con.close();
    }
}
```

c:\> javac CallableTest1.java

c:\> java CallableTest1

7788

Output : experience of 7788 is 24 years

Q) The following procedure accepts input as an employee number and returns bonus as output and it is calculated according to the salary of the given employee

[For this procedure code refer page no. 52 line num 261]

Q) For JDBC application to call the bonus procedure refer page 52 example-10.

→ While compiling this example we will go some more systems. Because, we used an old Java API called ~~java.io.InputStream~~ for reading input value at runtime.

→ While executing this procedure, we need to set input as a ~~execute query~~ number, we will get output as bonus. So we need few question marks while calling this procedure

Working with InOut parameter :-

→ If a procedure contains InOut parameter then in a JDBC application, we need the following two steps.

- ① Set Input value for that parameter
- ② Register the same parameter as output by calling `registerOutParameter()`

The following procedure accepts salary as input and returns salary as output.

Q) Create or replace procedure testpro (sal in out number) is

begin

sal := sal + 1000;

end;

endis

SQL> @ sample5

[Sample5 is a file]

The following JDBC program is for calling above procedure.

```

CallableStatement stmt = con.prepareStatement("call testproc(?,?)");
stmt.setInt(1, 8000);
stmt.registerOutParameter(1, Types.INTEGER);
stmt.execute();
int i = stmt.getInt(1);
System.out.println(i);
stmt.close();
con.close();

```

OUTPUT
9000

Syntax for creating a function is

Create or replace function functionname(parameters)

return returntype is

variable declaration;

begin

 stmts;

end;

The following accepts input of dept number and returns sum of salary of all employee working in that dept.

[For this function code refer page num 53 and line 90]

* In a function, all parameters are of type Int

* We can't write In keyword for parameters

* In a function definition, return stmt is mandatory

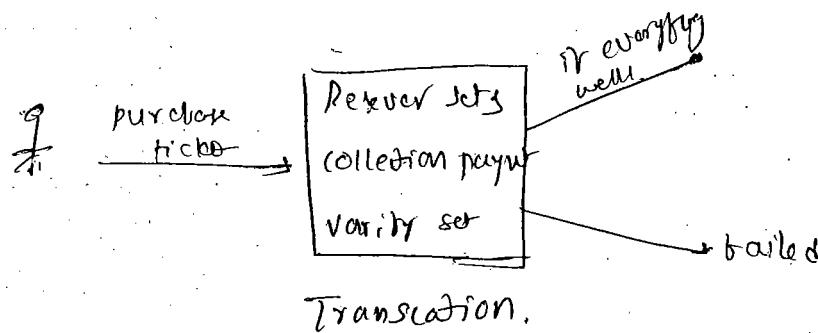
* While calling a function from a JDBC program, we use an additional step and it is for storing output value returned by a function

CallableStatement stmt = con.prepareStatement("call func()")

Date
8/8/11

Transaction Management in JDBC

- Defn: - A Transaction is a group of operations, made as single unit and produces a single outcome. This outcome is either success or failed.
- If all operations in a transaction are success or correct then that transaction reaches to success point.
 - If any one operation is failed failed then all the operations in the group are canceled and that transaction reaches to failed point.
 - For example in on-line ticket reservation if all operations are successfully then that transaction becomes success. If any one operation failed then all operations are canceled and the transaction become failed.
 - Each transaction follows a principle all or nothing



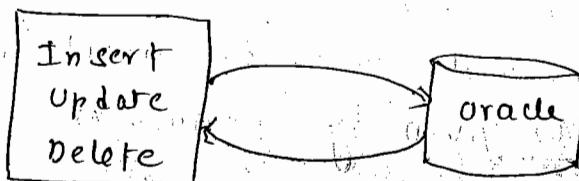
Types of Transactions

- ① Local Transaction
- ② Global or Distributed transactions
 - a) Flat transaction
 - b) Nested transaction

③ Local transaction :- If all operations are executed against one database then we called Local transaction.

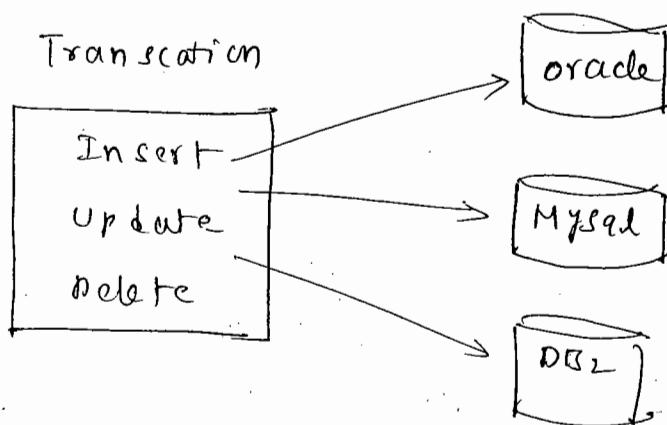
→ JDBC technology only supports local transaction

Transaction



Distributed transaction

- If all operations in a transaction are not executing against a single database that is executing on more than one database then it is called distributed transaction.
- EJB technology and spring frame work, both supports distributed transactions.



- By default, all operations that are executed on a database from a JDBC program are permanently executed, by default.
- The reason for executing operations on database permanently is, because in Java program (JDBC program), by default auto commit mode is enabled. So the commands are executed permanently on the database.
- If we want to apply transaction management in a JDBC program then the first step is we need to disable the autocommit mode.
- If we want to disable auto commit mode then we need to call a method setAutoCommit(false). This method is given by Connection interface.
- If any error occurs while executing a transaction on a database then we need to cancel all operations in

by calling rollback) given by connection interface.

- If no problem is occurred then we need to save all operation in the transaction as permanent on to the database. This can be done by calling commit() given by Connection interface.
- In a JDBC program, we need to put the transaction operation with on try and catch blocks before we start the transaction operations first if all AutoCommit mode must be disable.

Ex `con.setAutocommit(false);`

try
q

1

{ con. commit(); }

catch(Exception e)

१

(on, rollback);

1

* According to the above case, if any exception is accrued then the control entries into catch block and the truncation is canceled.

- * If no exception is occurred then the commit statement in the try block is executed. It means the transaction is successful.

Q) The following TDBC program to perform transaction management in TDBC. This application has three operations and if any one operation fails, transaction fails. If all operations are success then transaction becomes successful.

// Fibonacci Example.java

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

class JdbcTxExample

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

```
class.forName ("oracle.jdbc.OracleDriver");  
Connection con = DriverManager.getConnection ("jdbc:oracle:  
thin:@localhost:1521:satya", "scott", "tiger");  
Statement stmt = con.createStatement();  
con.setAutoCommit (false);  
try  
{  
    int k1 = stmt.executeUpdate ("insert into dept values ("  
        "99, IT, 'Hyd')";  
    int k2 = stmt.executeUpdate ("delete from satya where  
        sid = 111");  
    int k3 = stmt.executeUpdate ("Update emp set sal = 8780 where  
        empno = 7788");  
    con.commit();  
    sop ("Transaction success");  
}  
catch (Exception e)  
{  
    try  
{  
        con.rollback();  
    }  
    sop ("Transaction is canceled");  
}  
catch (Exception e1)  
{  
    sop (e1);  
}  
stmt.close();  
con.close();  
}
```

→ The following example tells transaction management method of JDBC using swing.

Source Code refer page no. 55 Ex - 19

Imp points

J

- In this transaction example, we used two static methods in the class
 - (i) main()
 - (ii) check()
- Both are static methods and both methods are exist in the same class. So we can call one method in another & method directly.
- In this example, connection is opened in the main() and it closed in the check(). So we created connection variable at outside the main()
- In Java, static methods allows only static variables of the class. So we made our connection variable as static.
- In order to get confirmation dialog box, we used a class called JOptionPane. We imported javax.swing package.
- To get confirmation dialog box, we used a static method of the class called showConfirmDialog().
- Our application either commits delete operation or cancels delete operation by depending on the option selected by the user.
- By default a frame is invisible. So, To make it as visible we called setVisible() and set width & height for frame we called setSize().

→ To exit the program execution whenever a browser is closed, we called a method setDefaultCloseOperation()

MetaData in JDBC

→ Data about data is called as metadata.

→ In JDBC 1.0, we have two metadata interfaces given in Java.sql package

(i) ResultSetMetaData

(ii) DatabaseMetaData

ResultSetMetaData :-

* When we select the data from a table or database, the rows selected from the table are stored in a ResultSet object. But to print that data, the programmer must known about that table structure.

* If the table is unknown table then it becomes complex to print the data from a ResultSet object.

* So in order to overcome above problem, JDBC has provided ResultSetMetaData interface under its rebase (object) will store metadata of the data stored in ResultSet. By using this Metadata object a programmer can bind or print the data stored in ResultSet.

* In order to get ResultSetMetaData, we need to call a method getMetaData() given by ResultSet object

* For example:

```
ResultSet rs = stmt.executeQuery("select * from xyz");
```

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

* According to the above code, the object rs contains data selected from the xyz table. and the object rsmd contains data about the data stored in the rs object.

* To get the meta data stored in rsmd object, we use the following methods.

→ ① getcolumnCount():
It returns no. of columns in each row of the data stored Resultset object. This method returns an integer value.

```
int x = rsmd.getcolumnCount();
```

② getColumnName(int column)

This method is for getting column name of the given column number. This method returns a String.

```
String s1 = rsmd.getColumnName(2);
```

```
String s2 = rsmd.getColumnName(x);
```

* Here s1 contains first column name and s2 contains last column name.

③ getColumnTypeName(int columnnum)

This method is for getting datatype of name of the given column. It also returns string.

```
String str1 = rsmd.getColumnTypeName(2);
```

```
String str2 = rsmd.getColumnTypeName(x);
```

* Here str1 contains first column datatype name and str2 contains last column datatype name.

① getColumnDisplaySize(int column)

It returns the datatype size for the given column. This method integer value.

```
int k = rsmd.getColumnDisplaySize(3);
```

→ For example of ResultSetMetaData refer page no. 53 exm 4/2

DatabaseMetaData interface

This is used to get data about database

→ This interface contains meta information like database name, version, driver name, version etc.

→ In order to get database metadata object, we need to call getMetaData() given by connection interface

Syntax :-

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

→ In order to get metadata information stored in dbmd object, we need to call the following methods

① getDatabaseProductName() :-

It returns database name

② getDatabaseProductVersion() :- It returns version of database

③ getDriverName() :- It returns jdbc driver name

④ getDriverVersion() :- It returns version of the driver

⑤ getMaxColumnsInTable() :- It return maximum no. of columns allowed while creating a table in the database.

⑥ getMaxTableNameLength() - It returns the maximum no. of characters allowed while writing a table name etc.

* Prop.

→ For this Database Metadata refer pg 53 ex-13.

New feature added in JDBC 2.0

- ① scrollable Resultset is included. By using this Resultset the cursor can be moved in any direction.
- ② Batch processing is added. It means the no.of round trips between Java and database are reduced or decreased.
- ③ Updatable Resultset. It means if do any modification in the Resultset object then automatically the changes are effected on the database table also.

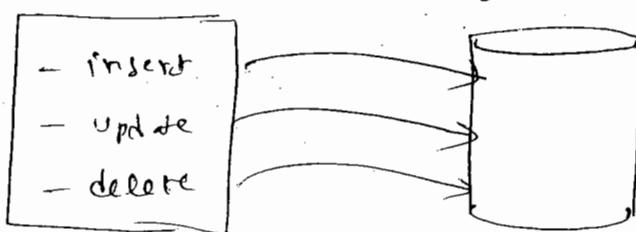
Date
11/7/11

Batch processing

- While executing multiple operations on database, one by one operation will be transfer on to the database at a time. It will increase no.of round trips between a Java application and a database.
- If no.of round trips are increased between a Java application and a database then automatically the performance of the application will be decreased.

Java application

Database



- In order to overcome the drawback of JDBC 1.0, in JDBC 2.0 the new feature got called as batch processing
- In Batch processing, it is possible to execute more than one operation on to the database at a time by making the operation as single batch
- Advantage of batch processing, is it reduces the no. of round trips between a java application and database so that performance of the application will be increased
- For batch processing, the following two methods given by Statement interface.
 - (i) addBatch()
 - (ii) executeBatch()
- addBatch() is for construction of a Batch
- executeBatch() is for execution of a batch on to database
- addBatch() is called for multiple times, in order to construct multiple commands as a batch.
- The entire batch is at a time executed on database, by calling executeBatch()
- The return type of execute() is executeBatch() by an Integer array. This array contains the integer values returned by the database, for each operation in the batch.

Ex:-

```
Statement stmt = con.createStatement();
stmt.addBatch("insert ---");
stmt.addBatch("Update ---");
stmt.addBatch("Delete ---");
int k[] = stmt.executeBatch();
```

- While executing a query, many ~~multiple~~ operations are involved that is insert, update and delete.
- select is not a batch processing operation.
- In batch processing we will never get a ~~resultset~~ so we need to pass type and more parameters for createStatement()
- While executing Batch on a database, if any ~~except~~ one operation is failed then the next operations in the are canceled and before operations are executed and an exception is thrown to the Java program may called as BatchUpdateException
- If we want to cancel total batch, if any exception is occurred then we need to apply transaction management along with batch processing
- For Batch processing without transaction management, refer 54 ex:- 16
- The following example is for transaction management along with batch processing

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

Updatable Resultset :- In this feature of JDBC 2.0, we may changes on Resultset. And the changes are automatically effected on database side.

- In this UpdatableResultset there is no difference between sensitive and insensitive Resultset.
- This JDBC 2.0 feature is only supported by type-2 JDBC driver. Other drivers are not supported.

3 Date
3 11/11

Connection pooling

→ Whenever a Java program wants to store the data (or persisting the data) in a database, first of all Java programs need a database connection. So we have the following two approaches

(i). Create a new connection and execute operation on database and then close it.

(ii). Use already opened connection and execute operation database.

→ In the first way, opening a new connection means taking some time. So performance of the Java programme will be decreased.

→ When using already opened connection then no need to wait for until connection is opened. Because connection is already opened.

→ In the second approach, there is a problem when multithreading is applied with synchronization. Here only one thread can use that connection and remaining threads have to wait. So it also effects performance of Java programme. So the solution for above two problems is a connection pool.

→ In connection pooling, a Java programme will ask for a connection and pool will give a connection to the Java programme. After the work is completed Java programme will give that connection again back to the a pool.

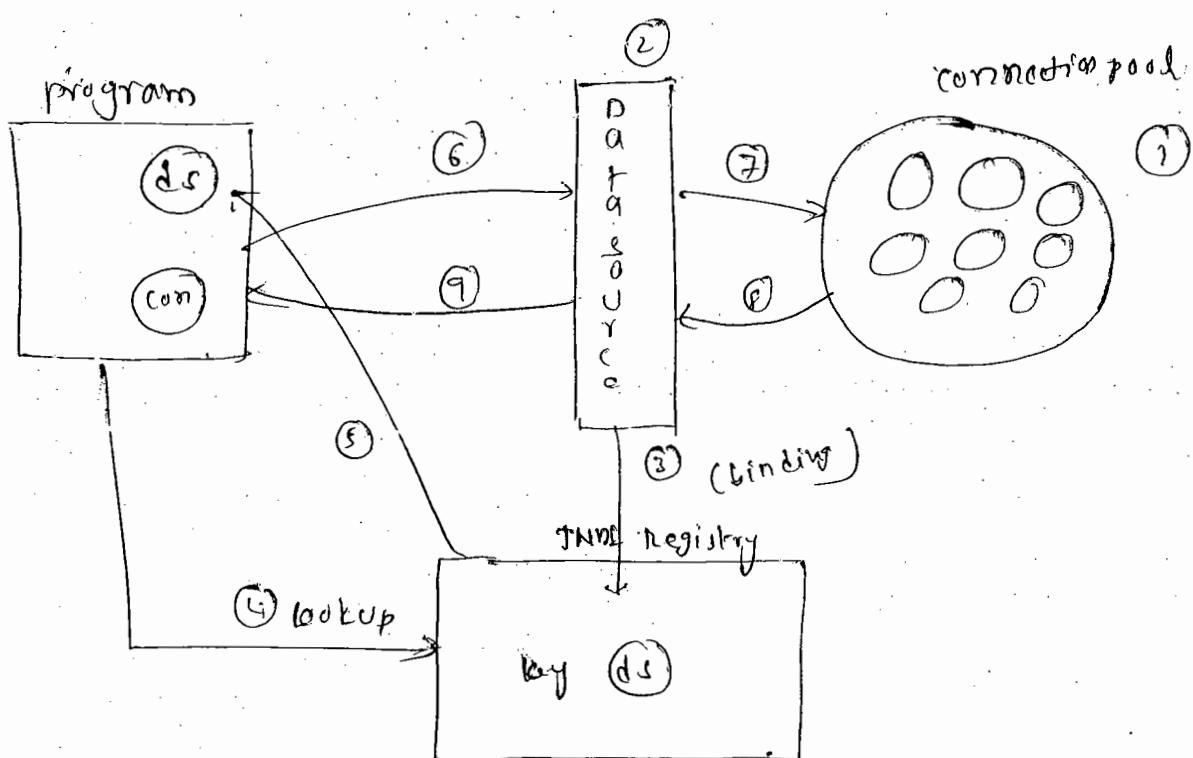
→ A pool contains some set of database connections, so at a time multiple threads can also use connection.

→ Connection pool is avoiding the problem in first approach and problem in second approach so the Connection pool is best solution while connecting with database from a Java application.

problems with DriverManager

- In JDBC programming, whenever we use `DriverManager` class it immediately opens a new connection with a database. After the work is completed then we need to close it.
- If suppose we are not closing that connection then that connection will become as a burden to the database server.
- `Driver Manager` always opens a new connection but it does not reuse the connections.
- If we want to open a connection with a database using `DriverManager` then we have to load the driver, we need to pass the url, usernames. so it increases burden on the client programmer.
- sun Micro systems given the a solution for a `DriverManager` called `Data Source Interface`.
- At present if a java programmer wants to obtain a database connection, JDBC API has given two ways
 - (i) Use `DriverManager` class given in `java.sql` package
 - (ii) Use `Data Source Interface` given in `javax.sql` package

→ In



connection pooling Diagram.

- In real time applications, server administrator and Java programmer both are different people so in real time there is a less burden on Java programmers.
- All real time applications of Java uses only connection pooling because in real time at a time multiple clients will send request to a web site or project.

With respect to the above diagram

- i) The administrator creates a connection pool with a database.
- ii) Administrator creates DataSource object. This DataSource object is mediator between Java applications and connection pool.
- iii) The DataSource object is binded (stored) into JNDI registry with some key. we call that key as a JNDI name.
- iv) A Java application uses JNDI API and performs lookup operation on the registry.
- v) Java application will get DataSource object back from the registry.
- vi) Java application now communicates with DataSource object.
- vii) DataSource object takes a connection from pool and creates a logical connection (proxy connection internally).
- viii) Finally DataSource object provides the logical connection back to the Java application.

Note:-

1. Whenever a Java application closes a logical connection then Data Source object will send the original connection back to the pool.
2. One connection pool contains one database related connections only. It is not possible to store multiple database connections into a single connection pool.

- An administrator can create any no. of connection pools, depends on the requirement.
- + If multiple databases connections are required, then multiple connection pools are required in the server.

Oracle ConnectionPoolDataSource class

- OCPDS is the class given in oracle.jdbc.pool package
- OracleCPS class is the combination of both connection pool and datasource implementation. It means whenever an object of this class is created then internally a connection pool and datasource both are created.
- This class is given by oracle for getting connection pool behaviour for a standard alone application
- By using this class we can get a connection from pool and we can create a logical connection required
- In connection pooling, by one physical connection at a time only one logical connection can be created. It means, after closing the first logical connection, it is possible to open a second logical connection.

Ex:- An example for creating connection pool with oracle in stand-alone application.

```
import java.sql.*;
import javax.sql.*;
import oracle.jdbc.pool.*;

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

```

OracleConnectionPoolDataSource ocp = new OracleConnection
PoolDataSource();

```
OCPI.setURL("jdbc:oracle:thin:@localhost:1521:fatya");
OCPI.setUser("scott");
OCPI.setPassword("tiger");

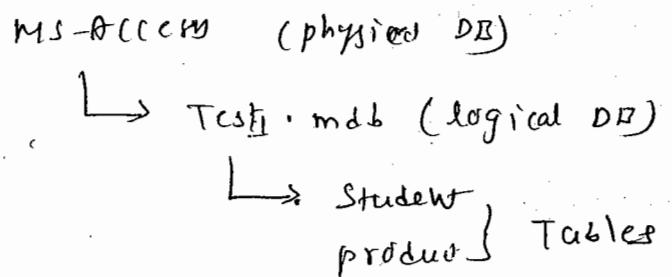
PooledConnections pc = OCPI.getPoolConnections();
Connection lcon1 = pc.getConnection();
Statement st1 = lcon1.createStatement();
st1.executeUpdate("create table sathyaliz(sno number(1), sname
SOP("table created"));
st1.close();
lcon1.close();
SOP("first logical connection closed");
```

```
Connection lcon2 = pc.getConnection();
Statement st2 = lcon2.createStatement();
int k = st2.executeUpdate("insert into sathyaliz values
(111,'sathya')");
SOP(" " + k + " row inserted ");
st2.close();
lcon2.close();
SOP(" second logical connection closed ");
```

3 To run above example we need set ojdbc14.jar in the class path

Connecting with MS-Access

- To connect with MS-Access from a Java application
There is only one driver supported called type-1
- MS-Access is a physical database we can create a logical database in that
- while creating DSN, we should select the logical database name



- while connecting with MS-Access no need of exchanging Username & password because authentication is not required

1. Open MS-Access and click on file → new → select blank database → Enter file name (Test1), save it our own drive → create
- Before creating the DSN, close the database Test1
- Start → Settings → control panel → administrator tools → Databases → Add Button → select Microsoft Access driver → Finish → Enter Data source name (AccessDB) → click on select button and select the drive and database → OK → OK → OK

211 //AccessTest.java

```
import java.sql.*;

class AccessTest {
    public static void main (String args) throws Exception {
        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
        DriverManager con = DriverManager.getConnection ("jdbc:odbc:access");
        System.out.println ("connection established");
```

```

Statement stmt = con.createStatement();
int ic = stmt.executeUpdate("create table Test (sno number,
sname text, marks number)");
if (ic == 1)
    System.out.println("Table created");
int ar = stmt.executeUpdate("insert into Test values (101, 'aaa', 300)");
System.out.println("1 row inserted");
stmt.close();
con.close();
}

```

Selecting data from a CSV file

We can use CSV file (comma separated value) as a database and we can select the data from it using JDBC API.

- we need to use type-1 driver for selecting the data from a CSV file
- while creating DSN, we need to select Microsoft Text driver
- A CSV file contains data of rows and each value in the row is separated with a comma (,)

sno	sname	marks
101	aaa	300
202	bbb	400
303	ccc	500

CSVTest.java

```

import java.sql.*;
class CSVTest
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection("jdbc:odbc:txtdsn1");
        Statement stmt = con.createStatement();
    }
}

```

```

ResultSet rs = stmt.executeQuery ("select * from test.csv");
while (rs.next())
{
    System.out.println (rs.getString(1) + " " + rs.getString(2) + " " + rs.getString(3));
}
rs.close();
stmt.close();
con.close();

```

Connecting with MS-Excel

1525 - oracle

MS-Excel (physical database)

↳ Book1.xls (logical database)

↳ sheet1
sheet2 } tables
sheet3

start → settings → control panel → administrative tools → data sources
 → add button → Microsoft xl driver → finish → enter DSN (addnew)
 → click on select work book → select sheet drive and work book name
 → de select read only check box → ok → ok → ok.

↳ rs = stmt.executeQuery ("select * from [sheet1\$]");

This is difference b/w Excel and Access

Connecting with MySQL

MySQL database vendor has provided a driver belongs to JDBC type-4 category called MySQL Connector/J (Java).

To get a connection with MySQL database, the following connection properties are required.

driver: com.mysql.jdbc.Driver
 package name class name

VRL: jdb:mysql://localhost:3306/test

// programme to connect with mysql

// mysqlcreate.java

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

```
class mysqlcreate
```

```
{
```

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

```
{
```

```
    Class.forName ("com.mysql.jdbc.Driver")
```

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

```
        "localhost:3306/test", "root", "1234")
```

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

```
    st.execute ("create table sathyajitho (sno INT, sname  
        CHAR (30))")
```

```
    System.out.println ("table created")
```

```
    st.close ()
```

```
    con.close ()
```

```
}
```

```
}
```

→ If we execute the above Java programme. Then we will get class not found exception, because mysql driver is unknown to the JVM.

→ We need to set driver related jar file in the classpath but this jar file doesn't come along with mysql database.

→ So we need to download a ~~JAR~~ file called MySQL-connector-jar file.

→ After extracting, in the root folder we will get the jar file related to the driver.

→ We need to set the jar to the classpath

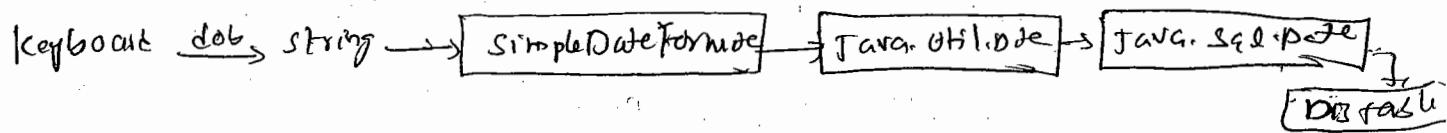
```
cd /usr/classpath = (: /mysql-connector-jar-3.0.8-stable/mysql-  
    connector-jar-3.0.8-stable-bin.jar; ) classpath %
```

- To verify whether table is created in MySQL or not
- click on start button → programs → MySQL Front → MySQL Front → connect(button) → expand(+) → find and verify the table

~~DD~~
17/11

Working with Date Values

- while working with databases from a Java application, the date format accepted by Java and the date format accepted by the database both are different
- While inserting or deleting ~~date~~ value using a JDBC application then we need some conversions between Java and database formats.
- In JDBC, Date class is given ~~in~~ Java.sql package, it acts as a mediator between Java date formats and database date formats.
- In Java, Date class is given in two packages
 - (i) java.util
 - (ii) java.sql
- From a Java programme if we want to insert date value into a database table. Then in the middle the following conversions are required.
 - (i) convert the Date value from string type into java.util.Date object
 - (ii) To do this conversion, we need a mediator class called Simple Date format class given in java.text package
 - (iii) convert java.util.Date object into java.sql.Date object
The java.sql.Date object will take care of converting the date value to database related formats
 - (iv) Now insert java.sql.Date object into database table



From String to Java. Util. Date object

Scanned Sz new Scanner (Systemin)

```
cop ("enter date of birth (dd/mm/yy yy);  
String str = s.next();
```

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
```

Java. Util. Date d1 = SimpleDateFormat.parse(str1);

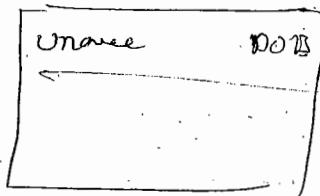
From java.util.Date (to) Java.sql.Date object

long ms = d1.getTime();

`java.sql.Date d2 = new java.sql.Date(m2);`

Setting data into database table :-

Prepared Statement part 2 (or, Prepare Statement Class)
into example values (1, 2) ??



```
pestools.setString(1, "abcd")
```

```
stmt.setDate(2, d2);
```

- ① The following JDBC programme is for inserting Username and Password example table of Database 9

```
sql> create table example (uname varchar(10), dob date);
```

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

import java.util.*;

smart favor. ten-^o

class Fibc Test

1

PS sum (String (in arr) flwg Elems)

1

```

class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:ordasq",
                                         "scott", "tiger");
PreparedStatement pmt = con.prepareStatement("insert into example
                                         values (?, ?)");
Scanner s = new Scanner(System.in);
System.out.println("Enter Username");
String str1 = s.nextLine();
System.out.println("Enter Date of Birth (dd-mm-yyyy)");
String str2 = s.nextLine();
SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
java.util.Date d1 = sdf.parse(str2);
long ms = d1.getTime();
java.sql.Date d2 = new java.sql.Date(ms);
stmt.setString(1, str1);
stmt.setDate(2, d2);
int rc = stmt.executeUpdate();
System.out.println("1 row inserted");
stmt.close();
con.close();

```

3

Output

c:\> java JdbcDataTest

Enter Username

abcd

Date of Birth (dd-mm-yyyy)

10-3-2009

1 row inserted

UNAM

abcd

10-MAR-09

say scott & brian example

Reading Date value from table

is whenever we read data from a table then we will get the data into resultset object

To read ~~the~~ Date values from Result object, we use getDates while reading, we have to convert

Java.sql.Date object back to Java.util.Date object

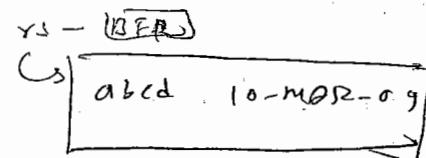
```
ResultSet rs = stmt.executeQuery("select * from example");
```

```
rs.next();
```

```
java.sql.Date d2 = rs.getDate(2);
```

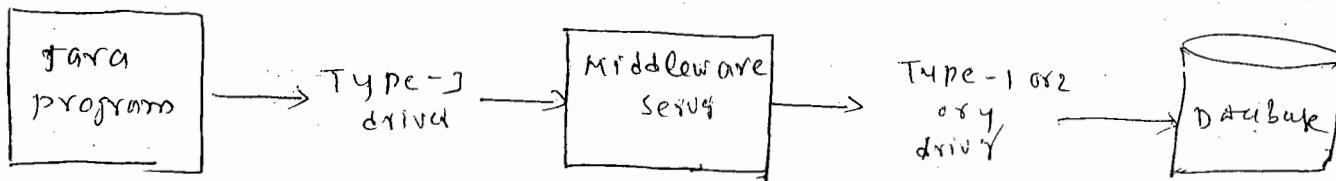
```
java.util.Date d1 = (java.util.Date)d2;
```

```
System.out.println(d1.toString());
```



Type-3 Net protocol pure Java driver

- This type-3 driver uses network protocol for connecting a Java-programme with a middleware server
- Type-3 driver follows three-tier architecture or three-tier model it means Java application connected with middleware server and middleware connect with Database
- The Type-3 drivers implemented 100% Java programming language so it is platform independent
- Type-3 driver connects Java programme with a middleware server but not the database directly so it is independent of databases



ex:-

Java programme → JDBC → JDS Server → Type-1 → DB

- In case of type-3 driver, the original database connectivity is taken care by middleware server
- Type-3 drivers will come along with middle ware servers. It means Type-3 drivers are server dependent.
- Type-3 driver doesn't take care about to which database the Java programme is connecting first it always connects Java program with middle ware server

Advantages:-

- Type-3 driver is both database independent and platform independent
- Type-3 driver is suitable for production environment (real time)

Drawbacks:-

- Type-3 driver is server dependent
- A middleware server needs to be installed to work with Type-3 driver
- Type-3 driver is slower than type-4 driver
- Type-3 driver uses a network protocol for connecting a Java programme with an application that is running at middle server. After that the application running at server will take care remaining database connectivity

Ex:- In case of, DDS server, JDBC driver connects Java programme or servlet application running in that Server. After that servlet application takes care about connecting with the database.

site:- DBSSoftware.com

IDB Server Information

- IDB SERVER IS open source Middleware Server, provides type-3 JDBC driver called IDB driver
- Whenever we installed the IDB server then it will be automatically started and it runs on port number 12
- We can get IDB server with from www.idbsoftware.com download.
- To verify whether IDB server has started or not, we can open windows services from administrative tools and we can find out in list of services IDB server started.
- The following Java program, represents Type-3 driver connection with oracle database using IDB server as middleware.

Program :-

```
import java.sql.*;  
class Type3Test2  
{  
    public static void main (String [] args) throws Exception  
    {  
        Class.forName ("idb.jdbc.IDBDriver");  
        Connection con = DriverManager.getConnection ("jdbc:idb://localhost:12/  
        /conn? driver=oraiden1&user='SCOTT'&pwd='TIGER'");  
        Statement stmt = con.createStatement ();  
        ResultSet rs = stmt.executeQuery ("select * from dept");  
        while (rs.next ())  
        {  
            System.out.println (rs.getInt (1) + " " + rs.getString (2) + " " +  
            + rs.getString (3));  
        }  
        rs.close ();  
        con.close ();  
    }  
}
```

- In the above Java program, we have used the memory
or don't. It must be.
- The server does not accept user input by connectivity.

Date

8/7/11

Introduction to Web applications

- In standard alone (desktop) two-tier application development mostly client-server applications are residing in a single system or probably different machines in a local area network.
- In standard alone two-tier application development, like Java socket programming (Networking) or JDBC programming, the client applications are thin clients or Fat clients.
- With standard alone client-server applications, we have the following two major problems.
 - (i) client application can't be accessed from anywhere in the network. It means on each system the client application need to be installed separately.
 - (ii) On one system, if a client application is modified then it will not be reflected on other systems where the client applications are installed, we need to go to each system separately and we need to make the changes individually. It is a time-consuming process.
- To overcome the above problems of standard alone Client Server applications, we moved on to client server web applications.

Web application :-

Whenever we want to provide service at application to a single client at a time then we can choose a standard alone application and which doesn't require any interfaces. When we want to provide service to multiple clients

Web application. Web application always runs on a central computer called as server.

Definition: A web application is a serverside application, which runs at server and provides service to multiple clients across internet simultaneously.

Types of web applications

(i) presentation oriented web applications

(ii) service oriented web applications

→ presentation oriented

- * These type of web applications are also called as static web applications.
- * These type of web applications will concentrate on providing static content on to the browser.
- * These type of web applications contains html files and image files.
- * The first type of web applications introduced are presentation oriented web applications.
- * presentation Oriented web applications do not accept any input from the client, rather web pages are displayed on to the client browser.
- * For example: An online tutorial is a presentation oriented web applications, because it contains static pages, which are already created or generated and one by one page will be shown to the browser whenever request is given.

(ii) Service Oriented

- * This type of web applications are also called as dynamic web applications.
- * This type of web applications will concentrate on providing dynamic services to the clients.
- * These type of applications are developed by using server technology like servlets, JSP's, EJB's, PHP etc.
- * In todays programming, almost all web applications are belongs to service oriented web applications.
- * For example Gmail, yahoo are Service oriented web applications and these websites will provided mailing and chatting services to the client across network.

Requirements for presentation oriented application

- > Browser software
- > Web Server software
- > Client side web technology
- >

Requirements for service oriented application

- > Browser software
- > Web server software
- > Client side application web technology
- > Server side web technologies
- > Database software

Introduction to web applications

⇒ In standalone (Desktop) two-tier application development mostly client & server applications are resided in a single system or probably different machines in a local area n/w's.

⇒ In standalone two-tier applications developments, Java ~~app~~ programming (a) JDBC programming the client appln's are "thick client" or "fat client".

⇒ with standalone server appln's, we have the following two major problems:

i) client applications cannot be accessed from anywhere in the network. It means on each system if client applications need to be install separately.

⇒ on one system, if the client app is modified then it will not be effected on other systems where the client appln's are installed.

and we need to make the changes. Individually it is a time consuming process.

→ To overcome the above problems in standalone client-server appl's so we move on to client-server web-appl's.

Q) What is web-application?

A- Whenever we want to provide service to single client at a time then we can choose a standalone applications and it doesn't require any internet access.

⇒ Whenever we want to provide services to multiple clients across network simultaneously then we need to choose web-application.

⇒ web-appl always runs on a central computer called a server.

Definition

A webapplication is a server side application, which runs at server side and provides service to multiple clients across at network simultaneously.

Types of webapplications:-

- (i) presentation webapplication.
- (ii) service oriented webapplication.
- (iii) presentation oriented web appn :-

This type of web-appn's are also called as static webappn's.

This type of web appn's will concentrate on providing static content on to the Browser.

→ These type of web applications contains HTML files & image files.

→ The first type of web applications introduced are presentation oriented web applications.

→ presentation oriented webapplications do not accept any input from the client, rather web pages are displayed on the client browser.

→ For example, an online tutorial is a presentation oriented webapplication because it contains static pages, which are already generated and one by one

(2) Get method appends form data in the address bar url whenever a request is submitted. if any secret data entered in the form it is also displayed in the address bar.

Get method is a Non-secure method.

(3) Get method doesn't support file uploading

(4) Get is an Idempotent method. it means get method goes to server and all the requested data

(2) When we submit a form post method do not effect form data in address-bar so post method is secure method than get method.

(3) post method supports file uploading.

it can not modify the data available at the server.

(4) post method is non-idempotent method, post method can modify the data available at server.

Note: post method can take the data from server and also it can modify the data available at server.

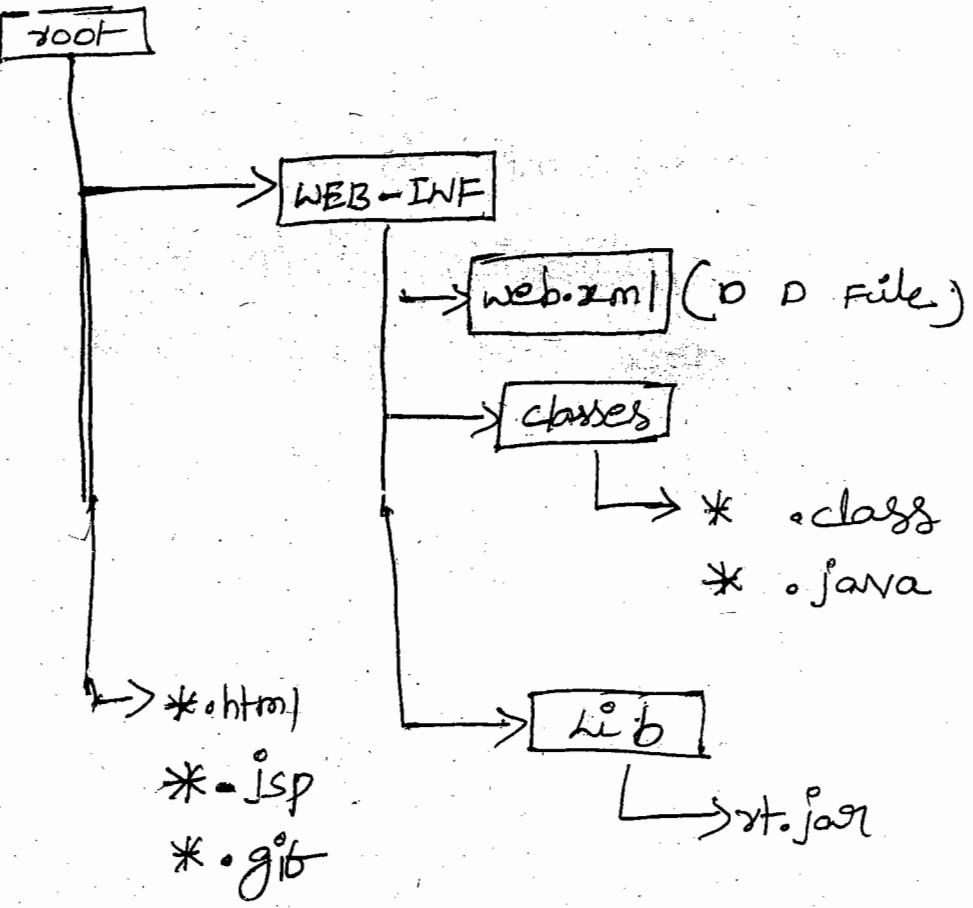
→ In an online exam, if we want to change from previous answer to new answer for a question then ~~is~~ HTTP-post method is required because, when answer is set for first-time, when we move on to the next question that answer will be stored in server.

→ when we come back to the previous question we change the answer & the previous answer in the server must be replaced with new answer for that question. In

→ In an interview questions website or a we
if the answer is not correct then new answer
can be typed by the user, so that at server
the previous answer is replaced with new
answer for that question. In this case HTTP
post method is required. because post is a
non-idempotent method.

Directory structure of a web-application

- While developing web-applications in Java, we must follow a standard directory structure given by SON (W3C).
- The advantage of following a standard directory structure is, our application can be executed on any server directly, it means our webapplication becomes server independent. By following the standard directory we will get a moto or feature (WODA) (Write once Deploy anywhere).



- In the above Directory structure root directory and its subdirectory WEB-INF or mandatory for a web applications.
- For each web-application one web.xml file is mandatory.
- ⇒ web.xml file is called as deployment descriptor file.

Deployment :-

deployment is a process of installing a web-application into a server. deployment can be done in following three ways.

- (i) Hand Deployment (ii) console deployment
- (iii) tool-based deployment.

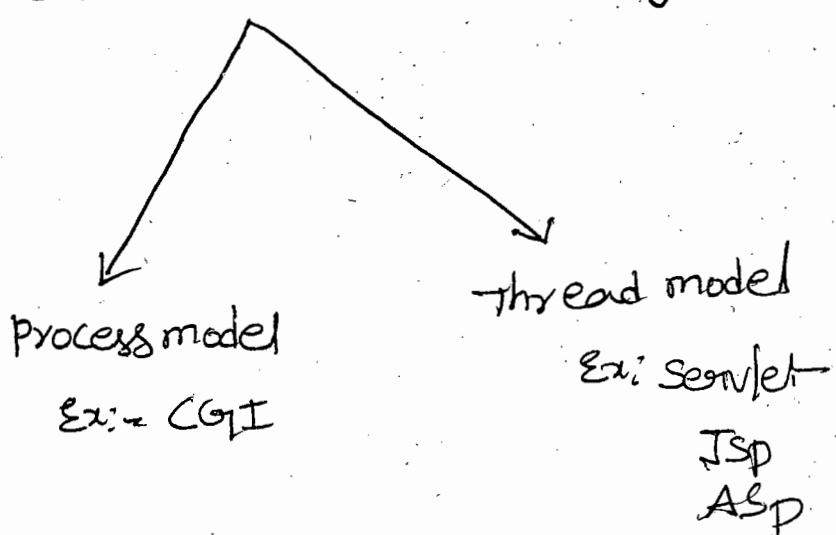
⇒ In Real-time application tool based tools are used (ANT, MAVEN)

↓
Another need tool.

why servlet Technology?

→ For developing server-side applications we have two models.

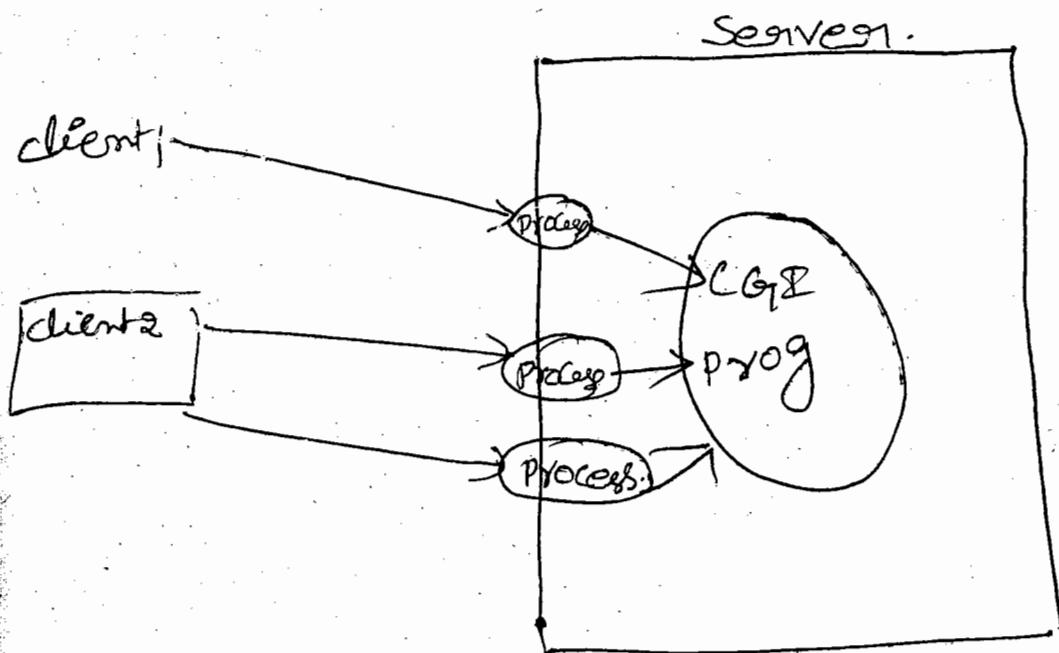
server-side programming



→ in the history of www (would by wide web) first technology introduced was CGI by NCSA

→ CGI technology follows process model, it means for each request a process is created at os level.

→ If the number of clients are increasing then automatically number of processors at os level are also got increased.



Drawbacks of CGI

- (i) CGI creates a new process for each request at server, so if the number of clients are increased or number of processes at server are also got increased. so

(ii) CGI uses PERL Scripting (Practically a reporting language) so CGI doesn't provide any security because security is not possible through scripting languages.

In order to overcome the drawbacks of CGI technology soon micro-systems introduced Servlet Technology.

22-7-14

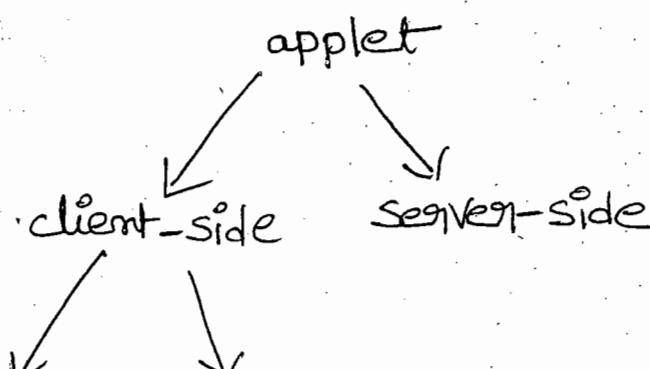
* What is Servlet?

In Java, applets are divided into two types

- (i) client-side applets.
- (ii) server-side ".

=> client-side applets are again divided into two types

- (i) un-trusted applet
- (ii) trusted



client-side applets in Java are called GUI applets or applets & server side applets are called servlet.

Applets are by default untrusted, it means an applet doesn't allow native code.

It is possible to convert an untrusted applet into a trusted applet by making changes in the applet policy file.

*** what is diff b/w an applet & servlet

Applet

- (i) An applet runs on a web-browser
- (ii) An applet increases (3) extends the browser functionality
- (iii) An applet provides service to one client at a time.

Servlet

- (ii) A servlet runs on a web-server.
- (iii) A servlet increases (3) extends a server functionality.
- (iii) A servlet provides service to multiple clients.

An applet
contains width &
height, it means
it visible on
browser, at many
it contains face.

4) A servlet doesn't
contain width &
height. It is
invisble in the
server, so a
servlet is face.

Def 1:

A servlet is a small, platform independent
java class it runs on a server & provides
service to the clients across network.

Def 2:

A servlet is nothing but, it is a server
side applet.

Def 3:

A servlet is a class which follows
a model called "single instance & multiple
threads".

Def 4:

A servlet is a dynamic web resource
of a web-application, which provides service.

Servlet technology:-

- Servlet is a Technology provided by Sun after C.G.I Technology, servlet technology given by S.U.N. for developing web Components.
- servlet Technology is a part of J2EE
- (a) JEE. J2EE has divided into following two types of Technologies.
- component Technology,
 - Servlet

Component Technologies

- Servlet
- JSP (Java Server pages)
- EJB (Enterprise Java beam).

⇒ servlet & JSP Technologies are used for web Component Development & EJB Technology is used for Enterprise component development.

Service Technology

- JNDI (Java naming & directory Interface)
- TMC (Java messas)

- (ii) Java Mail
- (iv) Java authentication authorization service
(JAAAS)
- (v) Java Connector Architecture (JCA).

Servlet API

Technology = API + Specifications

Servlet Technology containing both servlet API & servlet specifications.

Servlet API Contains set of classes & interfaces.

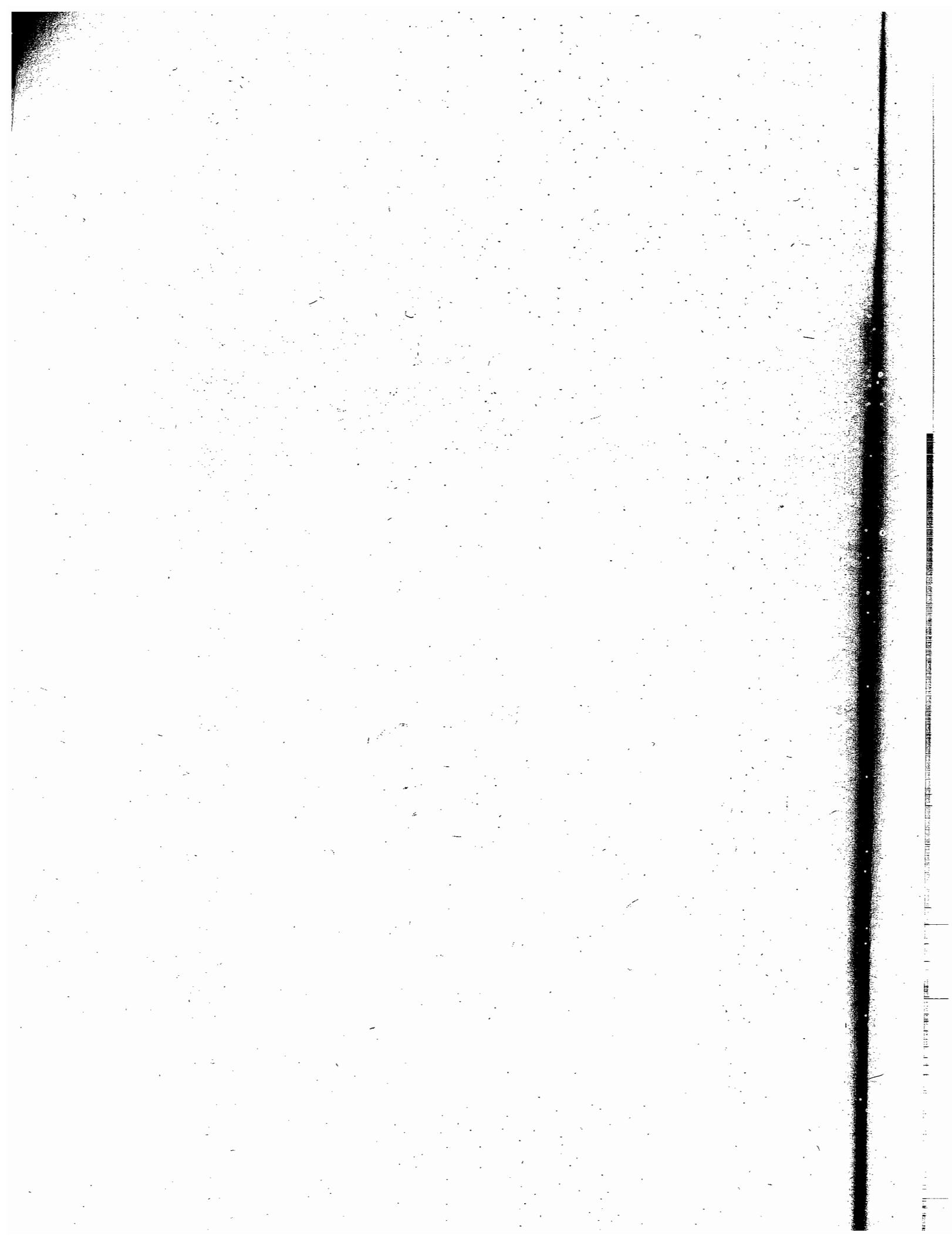
Servlet Specification contains set of rules & guidelines.

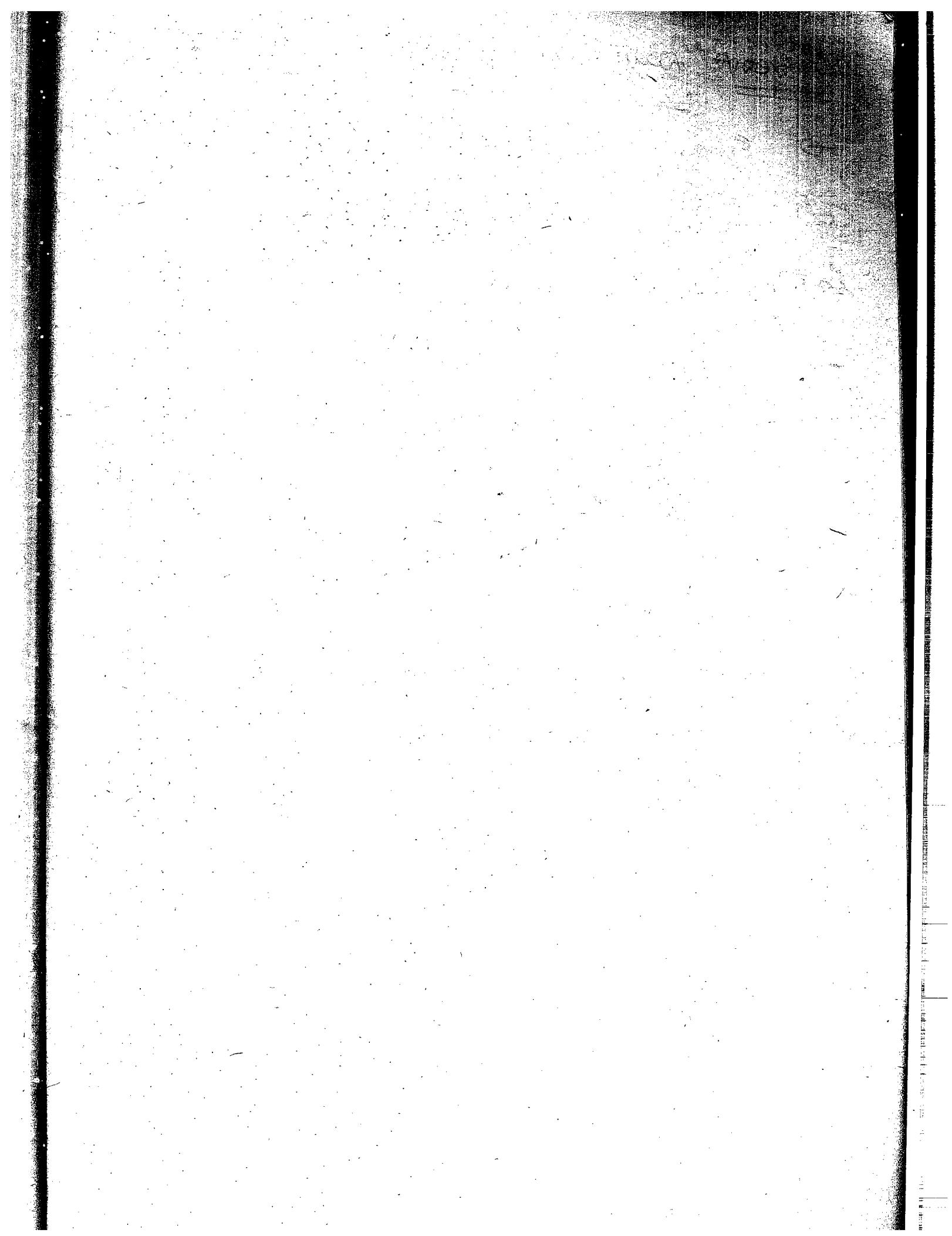
Servlet API consists the following two packages

javax.servlet;

javax.servlet.http;

- The servlet specification will be followed by the server vendor by providing implementation for interfaces of servlet API.
- Similar to servlet technology JDBC Technology also driver vendor will take care about implementation of JDBC interfaces.





Different ways of creating a Servlet.

- Every servlet class is either directly or indirectly implemented from `Servlet` interface given by `Servlet API`.
- For programmer convenience, `servlet API` has provided some abstract base classes, for creating a servlet.
- While developing web applications, a programmer can create a servlet class, by using any one of the following three approaches

(i) By Directly implementing our class from `Servlet` interface.

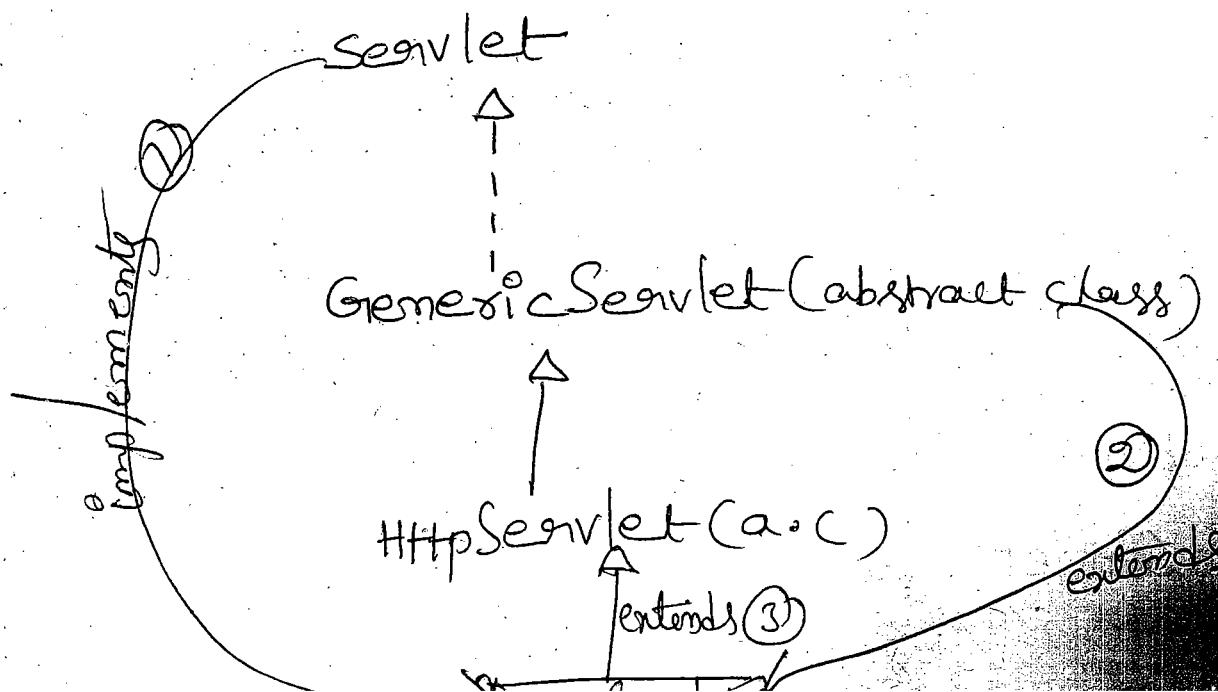
public class MyServlet implements Servlet { }

(2) By extending our Servlet class from a base class called GenericServlet.

public class MyServlet extends ~~Generic~~ GenericServlet
{
 //...

(3) By extending our Servlet class from a base class called HttpServlet

public class MyServlet extends HttpServlet
{
 //...



Note: (i) According to interviews point of view there is only one way of creating a servlet.

i.e., By either directly (or) indirectly implementing Servlet Interface.

⇒ (ii) While creating our servlet classes we assume that there are 3 ways of creating servlets.

(i) By implementing Servlet Interface

(ii) By extending GenericServlet

(iii) By extending HttpServlet

⇒ Servlet Interface and GenericServlet abstract class both are given in javax-

servlet package.

... HttpServlet is an abstract class given

no) javax.servlet.http package.

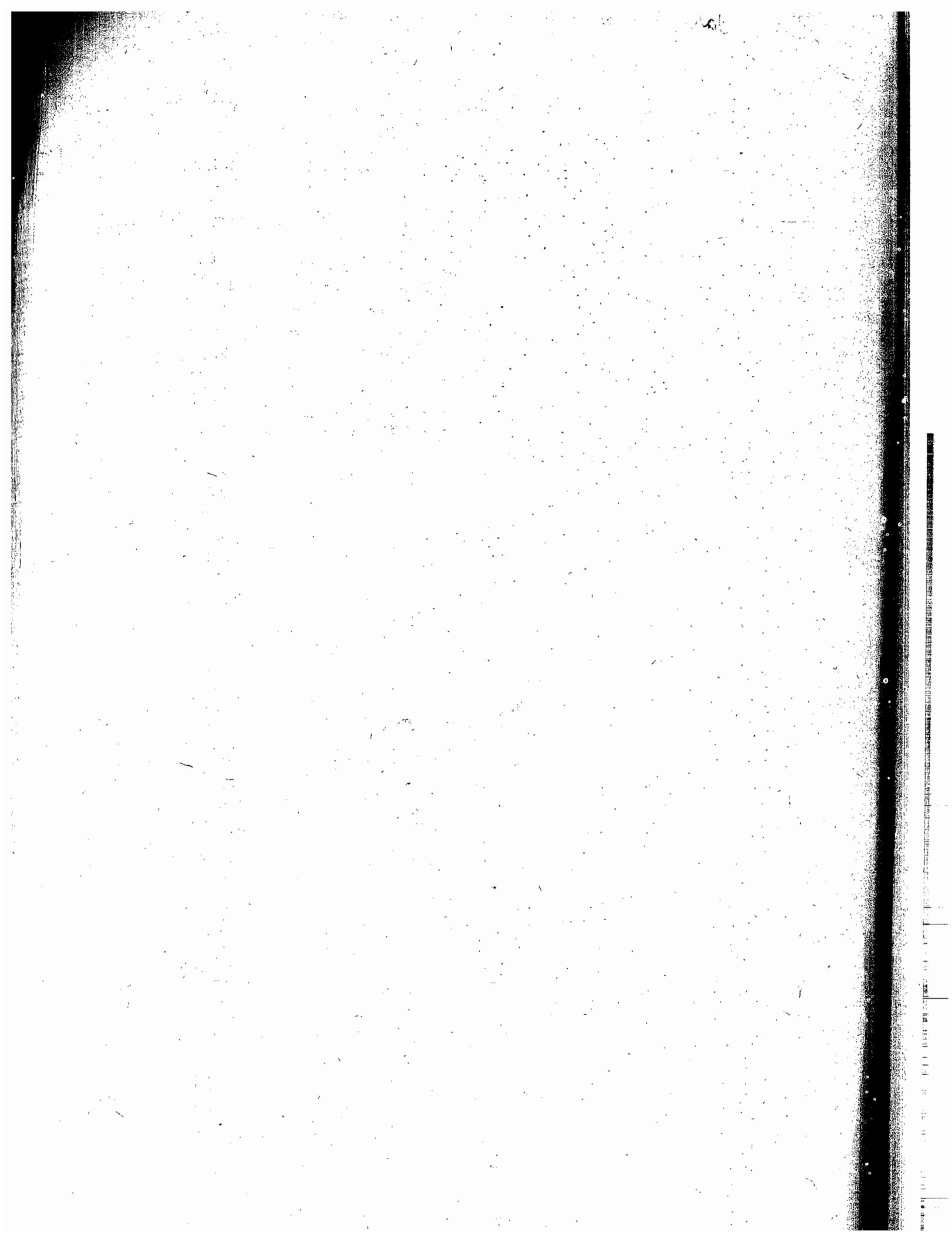
→ In Servlet API, GenericServlet is an abstract class, because it has one abstract method, but in HttpServlet class there are no abstract methods, but the class is given on abstract class.

life-cycle of a Servlet:-

- (i) Doesn't exist
- (ii) instantiation
- (iii) initialization
- (iv) Service
- (v) Destroy.

⇒ By default web container creates servlet object, whenever a request is given to the servlet.

⇒ Client has to wait until



a) when first reg is given
(\exists)

b) when Service is Starting \rightarrow Instantiation

Does not exist

Instantiation

a) construct(\exists)

init(\exists)

Initialization

a) when service shutdown
(\exists)

b) when Service is removed

Destroy

Destroy()

Service

service (regobj)
reg obj

→ After Servlet object created by the container, container initializes the servlet object by calling constructor of the class after that its life-cycle method called `init()`.

→ After a servlet object is initialized, A servlet object will receive the client request & performs processing & finally provides response back to the client.

→ we call these stage as service.

→ During service stage container calls another life-cycle method of a servlet called `service()` method.
 $(service(request, response))$

→ After processing all the requests which are given by the client, whenever the server is shutdown (3) A servlet is removed from the server then container will destroy the Servlet object

Container invokes

another life-cycle method of the Servlet called `destroy()`.

- whenever A ~~Servlet~~ object is destroyed it will be moved into again doesn't exist stage.
- Servlet life-cycle is a circular process it means, the life-cycle starts at doesn't exist & finally ends at doesn't exist.

~~Q:-~~ In what cases A Servlet object is Destroyed?

- A:-
- (i) when we shutdown the Server.
 - (ii) " " undeploy a servlet from a Server
 - (iii) when Server crash occurred.
 - (iv) when we stop the execution of a Servlet
 - (v) If Servlet is waiting for a long period of time in the Server.

Life-cycle methods:-

For a Servlet there are 3-life cycle methods and these life-cycle methods are provided by Servlet interface.

Servlet interface has totally provided 5 methods. & among 5 methods 3 methods are called life-cycle methods.

1) public void init(ServletConfig config)

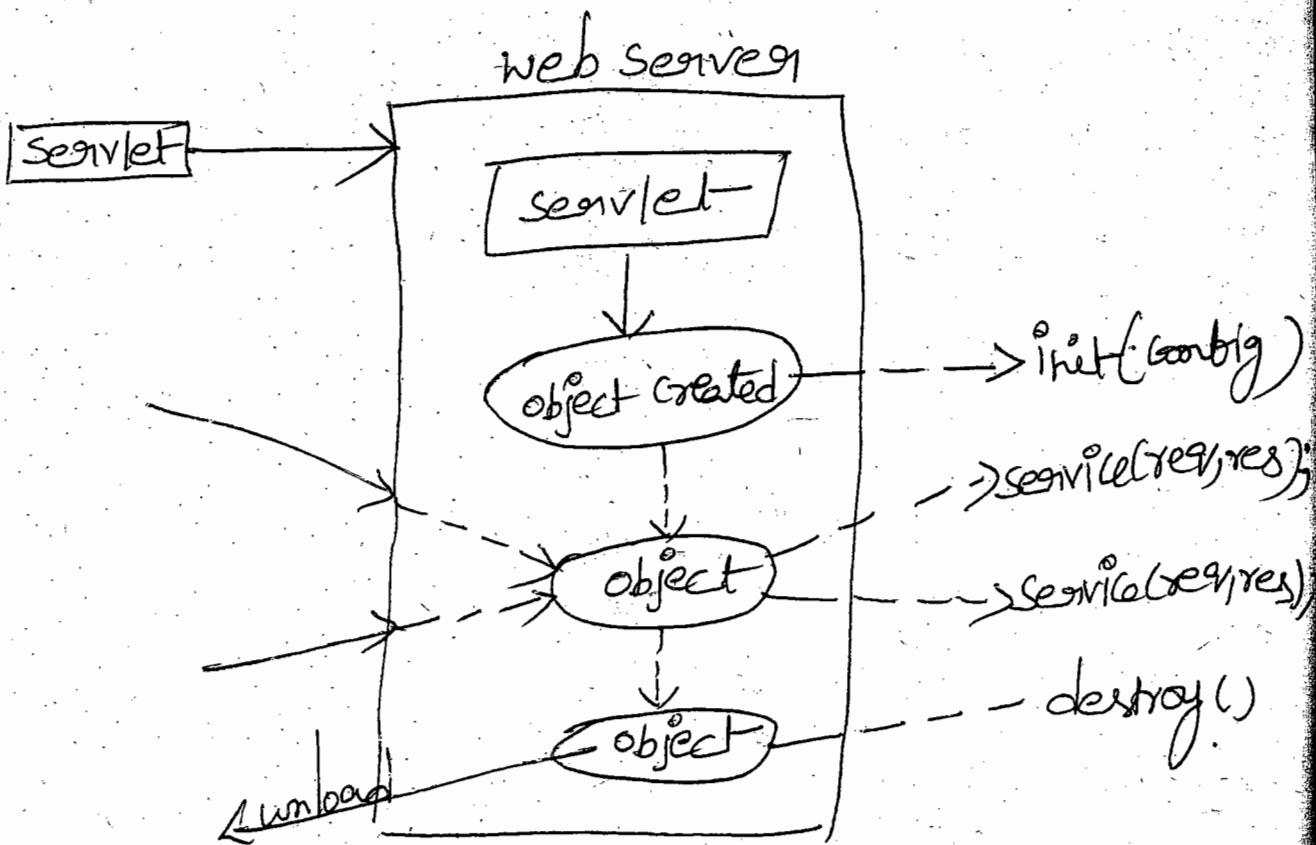
throws ServletException

2) public void service(ServletRequest request,
 ServletResponse response)
throws ServletException,
 IOException

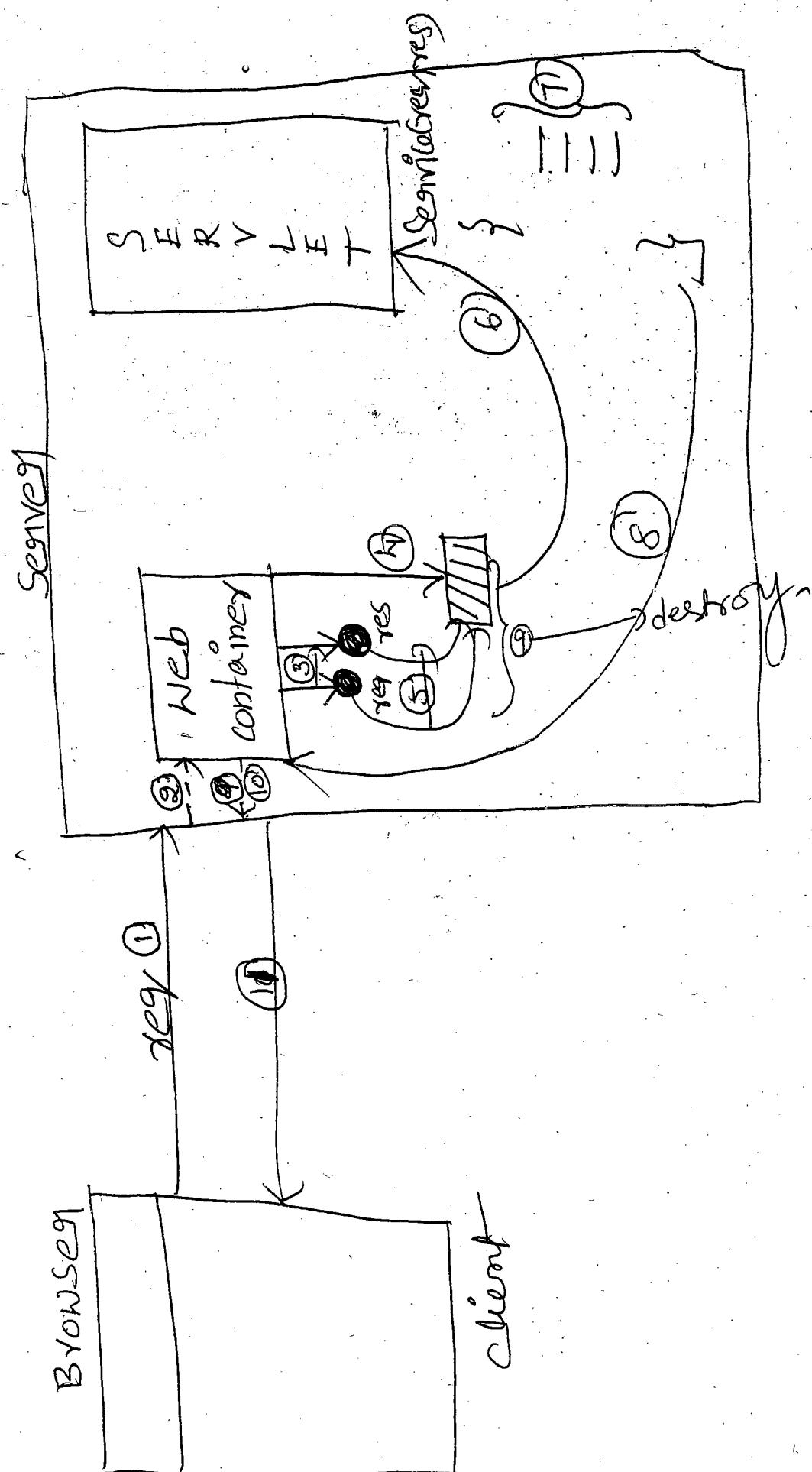
3) public void destroy();

→ init() & destroy() methods are executed only once, but service() method will be executed for each request.

The common way we use - you kernel of a servlet
is service method.



Request Processing in Servlet:



- (i) when a client request is given to the servlet, server received the request
- (ii) server verified that the request is for dynamic resource, so server transferred the request to web-container.
- (iii) container immediately creates a request object, response object & a thread.
- (iv) Container stores request objects into Thread object
- (v) The Thread will call service method of Servlet by passing request objects as parameters.
- (vi) The business logic implemented in service method is executed.
- (vii) The container will take the generated response from the servlet.
- (viii) The container will provide resp data to server
- (ix) The container will provide resp data to server
- (x) Container immediately Destroys the req, res obj, Thread obj

The server will construct the response from the form of a webpage then server will send the webpage data to the client.

Date

8/8/11:

class A

{

int x = 40;

class B

{

int x = 50;

void m()

{

cout < x; // How to print 40

}

}

}

Difference between `init(ServletConfig config)` & `init()`

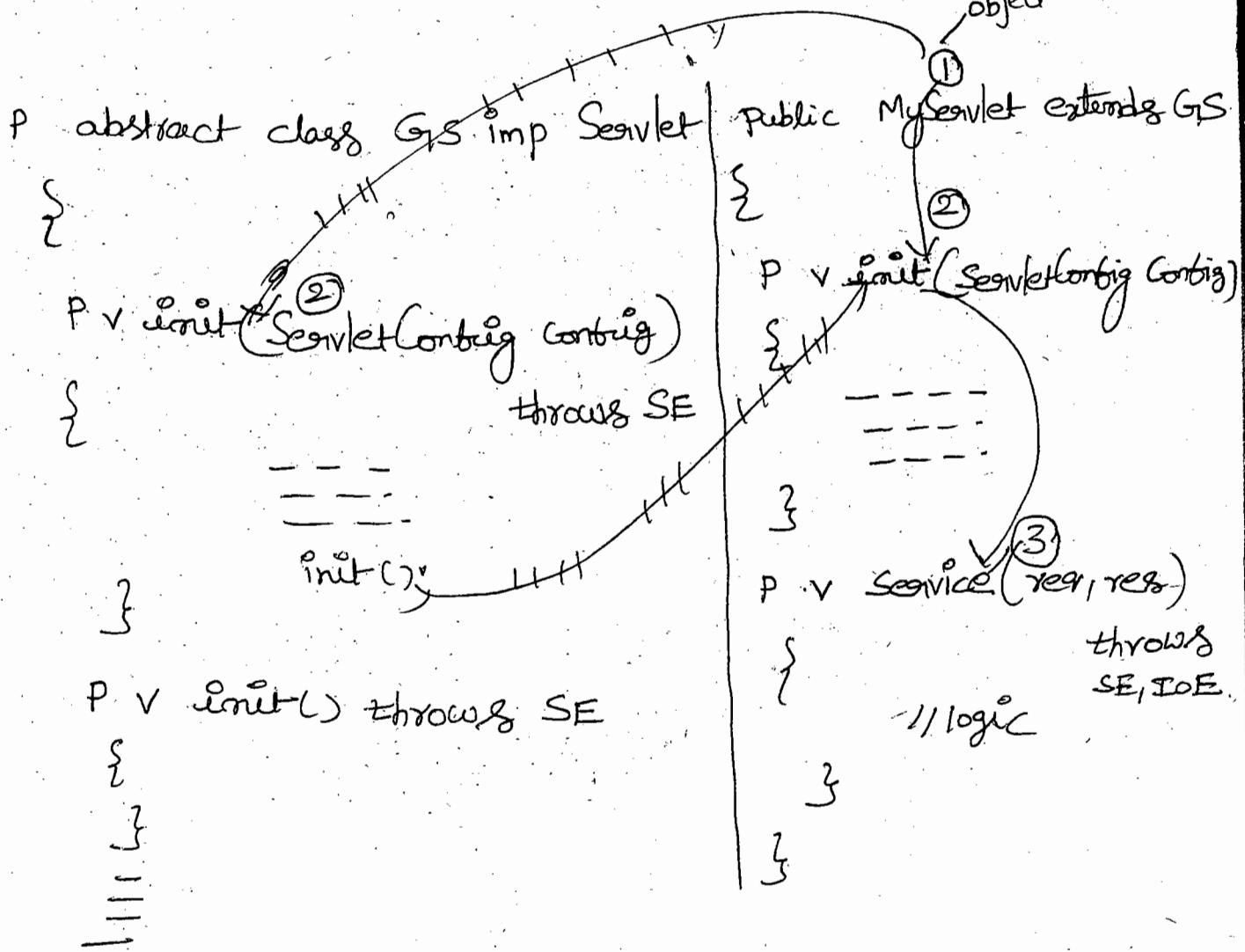
→ In GenericServlet class we have

two `init()` methods, where `init()` is a life-cycle method which is given by Servlet Interface & another `init()` is a non-life cycle `init()` added by GenericServlet and these `init()` doesn't contain any parameter.

→ In GenericServlet ~~the~~ class life-cycle `init()` method internally calls non-life cycle `init()`.

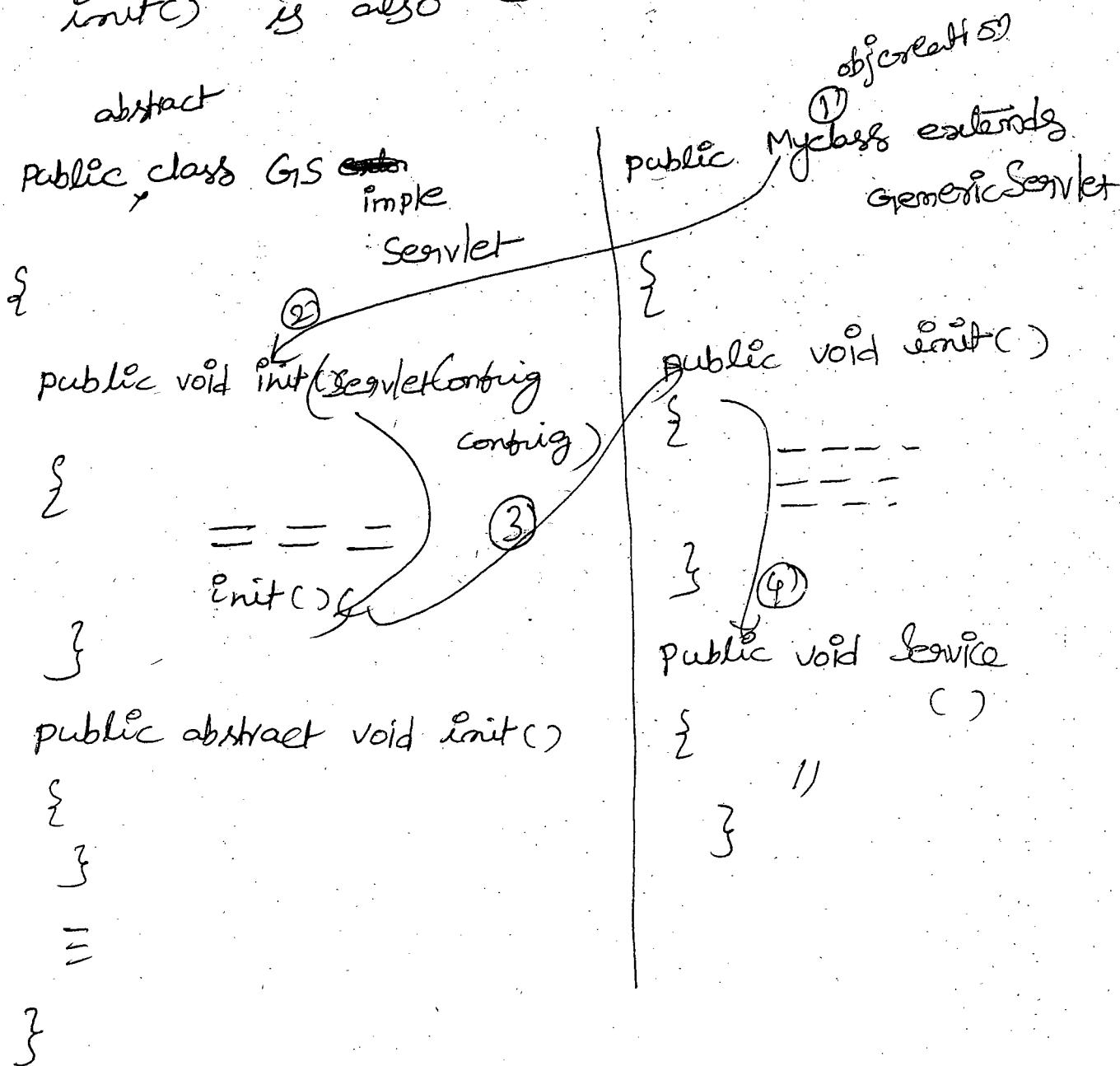
→ while creating our Servlet class, if we directly override life-cycle `init()` then container calls the `init()` available in our class & we are losing the logic implemented in the life-cycle `init()` of Base Class.

→ For example.



⇒ In order to overcome the above problem while creating our Servlet class instead of overriding life-cycle `init()` in our class we need to override non-life cycle `init()` given by GenericServlet class into our class.

The advantage of overriding non-virtual methods is we are not going to lose the base class implementation of the base class life-cycle `init()`. Base class life-cycle `init()` executed after that our class `init()` is also executed.



→ while overriding base class method in derived class low level access specifier replaced with same level or high-level access specifier.

→ High-level access specifier can not replaced with low-level access specifier.

→ For example.

P C MyServlet extends GS
{

 void init() ~~SP~~ throws SE // not allowed
 {
 =

 }
 P v service(req, res) throws SE, IOE
 {

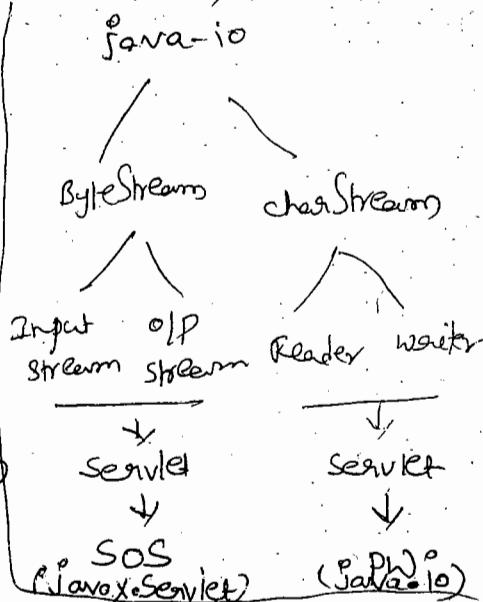
}

→ In the above servlet the init() is overriding concept so when we compile our servlet we will get an error.

The reason is, in the base class `fout()` is public & we are overriding with default access specifier, it means we are replacing high-level access specifier with low-level access specifier it is wrong.

* ServletOutputStream & PrintWriter classes

- In Servlet Programming, we have the two classes to generate the response i.e., (i) `PrintWriter`
(ii) `ServletOutputStream`.
- `PrintWriter` is a class from `java.io` stream & it belongs to character Stream category.
- `ServletOutputStream` is from `Servlet API` and it belongs to byteStream category.
- In `Servlet` class to generate response content we must need either `PrintWriter` class object or `SOS` class object.



- `PW` & `SOS` objects are provided by the

→ PrintWriter pw = response.getWriter();

ServletOutputStream sos = response.getOutputStream();
earmco

PrintWriter

ServletOutputStream

i) It is a character Stream.	ii. It is a Byte Stream.
(ii) PrintWriter is slow in network	(ii) It is fast in network.
(iii) It is only suitable for Text format of response.	(iii) It is suitable for both Text & Binary (Image) format of response.

Note:

If we want to send an image like gif, jpeg etc as a response back to browser from a servlet then we can not use PrintWriter, it is only possible through Servlet OutputStream.

Date
9/8/11

Extending HttpServlet

→ HttpServlet is an abstract class given in javax.Servlet.http package and it is extended from GenericServlet.

→ HttpServlet is also an abstract class and it has provided the implementation for the life-cycle service() method, which is an abstract method in the GenericServlet.

→ In HttpServlet class for providing service to the client requests, there are 9 methods.

→ In HttpServlet class, an additional service() method, i.e., non-life cycle service method is added and 7

doXXX() methods are added and all these methods are for providing service to the client request.

→ In HttpServlet class 2 init() methods are there but these 2 init() methods are not given by HttpServlet these are come from GenericServlet.

→ In HttpServlet class 2 service() methods are there one is life-cycle and other one is non-life-cycle service method.

→ In HttpServlet 7 empty doX() methods are given and these 7 methods are used to process the request transferred by 7 HTTP protocol methods.

→ For example, HTTPGet request, The processing method is doGet() and for post request the method is doPost() etc.

→ In HttpServlet class, its own methods added with protected access specifier.

public abstract class HttpServlet extends
GenericServlet

{

public void service(SR req, SRe res)

{

=

}

throws SE, IOE

~~public~~ void service (HttpServletRequest
protected req, HttpServletResponse
onse res)

{

throws SE, IOE

= =

}

protected void doXXX (HSSreq req, HSSres res)

{

}

throws SE, IOE

=

}

→ Why 7 doXXX() methods are given has
Empty methods in Servlet why not abstract?

A:- If HttpServlet developed provided
seven doXXX() methods has abstract them
when a programmer is extending the
class from HttpServlet, programmer has to
override all 7 methods are mandatory, it
increases the burden on a servlet
programmer. In order to reduce the burden
on servlet programmer, This 7 methods are
given has Empty but not abstract.

→ While creating our Servlet class by exten-
ding HttpServlet, we mostly prefered to
override either doGet() method or doPost()
method. because HTTP protocol mostly uses
either Get() or post() method to transfer
the request.

public class MyServlet extends HttpServlet

{

 public void doGet (HttpServletRequest req, HttpServletResponse res)

 throws ServletException, IOException

{

==

}

 public void doPost (HttpServletRequest req, HttpServletResponse res)

 throws ServletException, IOException

{

==

}

}

Note :-

In HttpServlet class the doXXX() methods are Protected and while overriding in our class either we can use protected access specifier or public access specifier.

- In the above servlet, it gets Request is given them doGet() method is call, then post request is given doPost() method is call.
- doGet() & doPost() methods are non-life cycle methods so a web-container doesn't call them directly.
- The client request reaches to either doGet() or doPost() in the following way
 - (i) when a client request is given then container calls life-cycle service() method.
 - (ii) life-cycle service method converts ServletRequest into HttpServletRequest and ServletResponse into HttpServletResponse and then calls non-life-cycle service() method.
 - (iii) non-life-cycle service() method verifies client request is given get() method or not and then calls the doGet()

public abstract class HttpServlet - extends GFS

{

public void service(ServletRequest req, ServletResponse res)

(2)

throws ServletException

}

req ---> hreq

res ---> hres

service(hreq, hres);

}

(3)

protected void service(HttpServletRequest req, HttpServletResponse res)

{

throws ServletException

(4)

// read the http method from the given request

if (method is GET)

 we call doGet(req, res);

else if (method is post)

 calls doPost(req, res)

}

==

class MyServlet extends HttpServlet

{
 ①}

 public void doGet(HttpServletRequest req, HttpServletResponse res)

{

 }

 throws ServletException

}
 public void doPost(HttpServletRequest req, HttpServletResponse res)

{

 }

 throws ServletException

IOException

3

**

Q:-

Why HttpServlet class given has abstract methods even though there are no abstract methods in it?

A:- If HttpServlet class is not given has abstract then A servlet programmer can create any number of objects for its directly. It violates Servlet principle or model.

so in order to disallow servlet programmers to extend the class

Made as an abstract class.

Because the class is an abstract class it is possible to ~~is~~ only extend it not possible for it to create object directly.

Date 11/8/11

⇒ If a servlet programmer is unknown about HTTP method i.e., either get or post. then while creating the servlet the programmer has to add doGet() and doPost() methods in the class with logic.

⇒ Instead of defining the logic both doGet() & doPost() methods we can reduce the size of servlet class in the following 2 ways.

(i) Call an user-defined method from both doGet() & doPost() method.

public class MyServlet extends HttpServlet

{

 P v doGet (req, res) throws SE, IOE

{

 process (req, res);

P v doPost(req, res) throws SE, IOException

{

process(req, res);

}

P v process(req, res) throws SE, IOException

{

//

{

(ii) Call either ~~doPost~~ doPost() from doGet()
or vice versa.

public class MyServlet extends HttpServlet

{

P v doGet(req, res) throws SE, IOException

{

doPost(req, res);

{

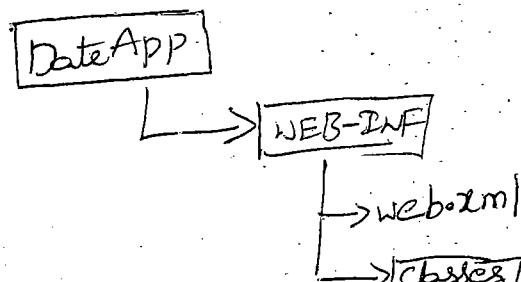
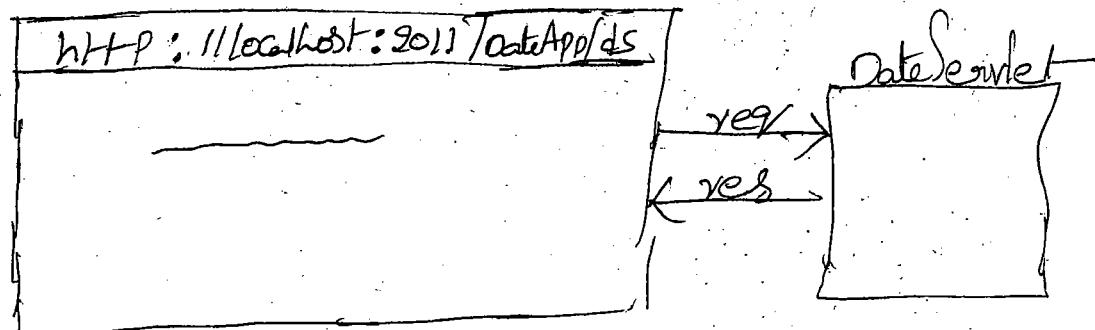
P v doPost(req, res) throws SE, IOException

{

==

{

The following web-application is to display Date & Time from servlet on to the browser and to refresh the response for each second?



DateServlet.java

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

```
public class MyClass extends HttpServlet
```

```
{
```

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

```
{
```

```
Date d = new Date();
res.setHeader ("Refresh", "1");
pw.println("<h2>");
pw.println(d.toString());
pw.println("</h2>");
pw.close();
```

{

}

②

web.xml

⇒ when request is given directly from a browser to a servlet always request is transferred by using `HttpGet()` method. so in our `Servlet` class we have overridden `doGet()` method.

⇒ to refresh the response of a servlet automatically on to the browser we have set a header in the `Servlet` the `response` or `response object`.

→ For the above Servlet we have added the "Refresh" value 1, which means Response will be automatically refreshed for each second.

* MIME SETTINGS:-

Before displaying a servlet generated response on to the browser, the web-server will construct a web-page and transfer that web-page on to the browser.

- If we don't set any MIME (Multipurpose Internet mail Extension) then by default MIME type is considered text/html.
- If we explicitly set MIME type to the response data then the server will construct a web-page of that type, and server will send that page on to the browser.

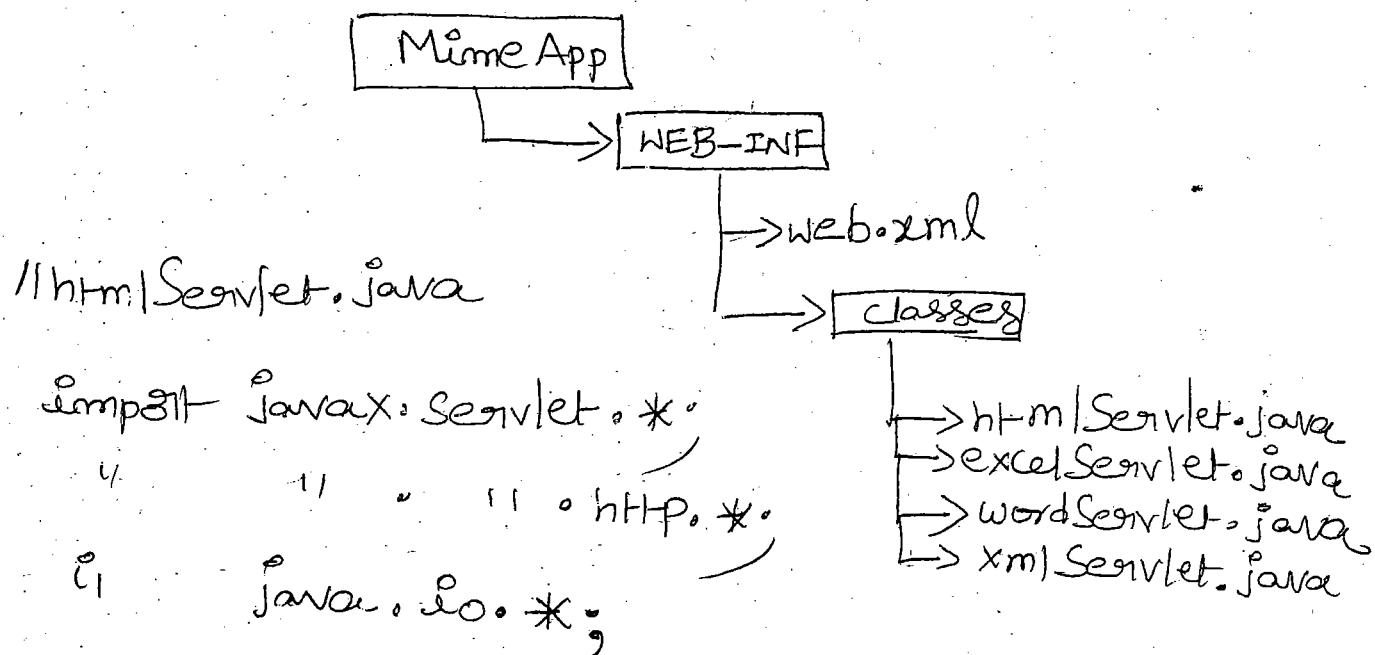
Some of the MIME types given by World Wide Web Consortium (W3C) are:

- | | |
|--------------|----------------------|
| ① text/html | ⑥ application/msword |
| ② text/plain | ⑦ " /vnd.msexcel |
| ③ text/xml | ⑧ " /pdf |
| ④ image/gif | ⑨ etc... |
| ⑤ image/jpeg | |

→ If we want to set MIME type explicitly we need to call setContentType() method which is given by ServletResponse interface

response.setContentType("text/plain");

* The following web-application is for displaying response using diff MIME types



P C HtmlServlet extends HttpServlet

{

 public void service (HttpServletRequest req,
 HttpServletResponse res)

{

 throws ServletException

 res.setContentType("text/html");

pw.println ("table border=2");

pw.println ("<tr><th> Name </th><th>");

Name </th></tr>");

pw.println ("<tr><td> 101 </td><td> venu </td>");

pw.println ("<tr><td> 102 </td><td> siva </td>");

pw.close();

}

}

// ExcelServlet.java

res.setContentType("application/vnd.ms-excel");

// wordServlet.java

res.setContentType("application/msword");

// XMLServlet.java

res.setContentType("text/xml");

pw.println ("<table>");

web.xml

<u-p> /host </u-p> to1. htmlServlet
<u-p> /evsl </u-p> " excel "
<u-p> /wvsl </u-p> " word "
<u-p> /xvsl </u-p> " XML "

Basic steps Console Deployment in Tom-Cat

Step 1:

Create a war file (webarchive).

Step 2:

In Java, we have the following three types of compressed Components.

- a) jar --- Java archive
- b) war --- web archive
- c) ear --- enterprise archive.

To create the archive components, we need to use a tool given by jdk called jar tool.

In web-applications, we need to create a war-file and it is a compressed format of the entire web-application.

It is similar to Win-Zip.

To create a war file, we need to place the .war file into the root directory of the web-app.

c:\LoginApp> jar cvf logtest.war *

c->create, v-verbose t-trile.

ear=jar+jar
ear=jar+war

Step 3:

Start Tomcat Server.

Step 4:

open the browser window & type the following url.

http://localhost:2011

Step 5:

click on Tomcat Manager

Enter usename
↓ Tab

password

↓

Go to war file to deploy section

↓

click on Browse

↓ Select war file

↓ click Deploy

Step 7:

open the Browser & type the following url

In address bar.

http://localhost:2011/logtest/login.html.

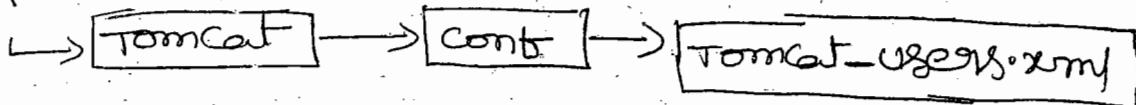
↓

Note:

After deploying the war file if we get running state false then there is a mistake in web.xml so we need to correct web.xml

→ If manager username or password is unknown then we can bind it through tomcat-user.xml

C:\



String s = "Satya" + "100" + "Tech"

How many objects created

only one object created because whenever arithmetic operator (+) is occurred string is converted into StringBuffer, so StringBuffer rule

Methods to read Url information

→ When sending a request to the server

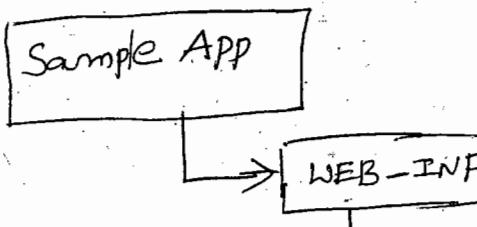
from client browser, we type some url in the address bar and it contains a diff part

→ For HTTP Servlet Request The following methods are available to read diff parts of the url.

- 1) `request.getScheme()` → returns protocol name like http, https-
- 2) " `getServerName()` → returns server machine Ip Address used by the request
- 3) " `getServerPort()` → Returning port number of the server
- 4) " `getQueryString()` → Returning query string added to the request.
- 5) " `getRequestURI()` → returning uri of the request (context path)
- 6) " `getRequestURL()` → ServletPath
(url) Returns complete url.
- 7) " `getRemoteAddr()` → Returns ipaddress of the machine who

Note:

- `getHttpServletRequest()` method returns `String`
- and all the Remaining methods return `String`
- * The following web-application is to print the different parts of the url of a request.



```
import javax.Servlet.*;  
import javax.Servlet.HttpServlet*;  
import java.io.*;
```

```
public class SampleServlet extends HttpServlet
```

```
{
```

```
    public void doGet(HttpServletRequest req,  
                      HttpServletResponse res)
```

```
        throws SE, IOException
```

```
{
```

```
    process(req, res);
```

```
}
```

```
    public void doPost(HttpServletRequest req, HttpServletResponse res)
```

```
        throws SE, IOException
```

```
{
```

```
    process(req, res);
```

P. v process(HSR req, HSRP resp) throws SE, IOE

{

Servlet output Stream SOS = response.getOutputStream()
Stream()

SOS.println("Method:" + request.getMethod())

SOS.println("
");

SOS.println("QueryString:" + req.getQueryString());

SOS.println("
");

SOS.println("RequestURI:" + req.

SOS.println("
"); getRequestURI())

SOS.println("RequestURL" + req.getRequestURL().toString());

SOS.println("
");

SOS.println("RemoteAddress:" + req.getRemoteAddr());

SOS.println("
");

SOS.println("Server Name:

" + request.getServerName());

SOS.println("
");

SOS.println("Server port:" + request.get

(int) + "
"); ServerPort());

```
sos.println("Servlet Path: "+req.getServletPath());  
sos.close();  
}  
}
```

http://localhost:2011/SampleApp/SameSrv?username=
O/p method: GET user=Satyam &
age=10.

queryString: user=Satyam & age=10

RequestURI: /SampleApp/SameSrv

Request URL: http://localhost:2011/SampleApp/SameSrv

Remote Address: 127.0.0.1 (st-andalone or desktop
systems (i.e., those

systems not
connected to
the internet))

Server name: localhost

Server port: 2011

Servlet path: /SameSrv

Q:- How to reject a request given to a
Servlet from a particular client system?

A:- If we want to reject client request
given from a particular system to a Servlet,
then we need to find IP address of the
client machine, then we need to reject the

→ To find IP Address of a System we call
getRemoteAddr() method.

for ex:

```
String s = req.getRemoteAddr();
```

```
if(s.equals("10.15.12.7"))
```

```
pw.println("Or request is rejected")
```

Q: How to reject request given by HTTP,

or FTP protocol?

A:- we need to call get-scheme method
to read the protocol name, from the given
request and then we need to verify
the protocol to reject the request.

for ex:

```
String s = request.getScheme();
```

```
if(s.equals("http")) || s.equals("ftp")
```

```
pw.println("Or request rejected")
```

~~Dev~~
15/8/11 General purpose IDE without Supporting Java
~~E~~E capabilities

Myeclipse = Eclipse + plugins

→ ADDED JavaEE

capability

⇒ in eclipse all works are done by manually.

⇒ The diff b/w Eclipse & edit - plugin

is in Eclipse we have Auto-compilation,
Automatic setter, getter methods.

⇒ NO Expiration time for Eclipse.

My-eclipse trial period - 1 month

web-application Development in My-eclipse6.0

(i) My-eclipse IDE (Integrated Development Environment)

as a Java-related J2EE Enabled IDE and it provides Rapid-application development

J2EE applications.

(ii) Eclipse is a General purpose IDE

and it doesn't contain inbuilt, J2EE

(iii) my-eclipse is created by adding plugins to the eclipse IDE.

my-eclipse = eclipse + plugin.

→ with Myeclipse IDE it is possible b/c as to develop, deploy & test a web-application within the IDE window.

→ Myeclipse is the most popular IDE used by Java-developer in the real-time environment than other-IDE's.

Step 1:-

Start →

programs →

myeclipse-6.0 → myeclipse-6.0

Enter workspace-name

D:\> venu\Servlets

any user
defined workspace

OK

Step 2:

File →

New → project →

webProject → Next → Enter →

Proj name (TestApp) → Finish

Step 3:

At left side we will get package explo
r window and in this window classes
folder is invisible under WEB-INF. So
shift from Package Explorer window to
navigator window.

window → show view → other → ~~Navigator~~
 (Expand)
 Navigator → Navigator. → Ok.

Step 4:

Right click on project Name → New → Servlet
 → Enter name (Servlet-class Name) → Next →
 Enter * Servlet-name (logins) & Mapping-URL (/logins) →
 Finish.

Step 5:

insert the required servlet-logic, either in
 doGet or in doPost() methods.

Step 6:

```

  {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String s1 = request.getParameter("uname");
    String s2 = request.getParameter("pwd");
  }
  
```

out.println ("Login success")

else

out.println ("Login not success")

Step 6:

⑩ Right click on projectname

↓
new

↓
HTML

↓
Enterprise Beans
(login.htm)

↓
Finish.

<form action="logsrv(url-path)">

username: <input type="text" name="uname">

password: <input type="text" name="pwd">

<input type="submit" value="click">

</form>

Step 7:

click on window menu → preferences → MyEcl

ipseCat (left-side) Expand it → Application Server →

tomcat → tomcat 6.0 → select enable →

click on browse button → select tomcat root directory
(tomcat 6.0)

→ apply → ok

Step 8:

- Right click on project name → Myeclipse
- Add & Remove Project Deployments → Select project name (TestApp) → Add button → Select server name (Tomcat 6.0 X) → Finish → OK
- Tomcat 6.0 X → Finish → OK

Step 9:

click on down arrow of myeclipse servers

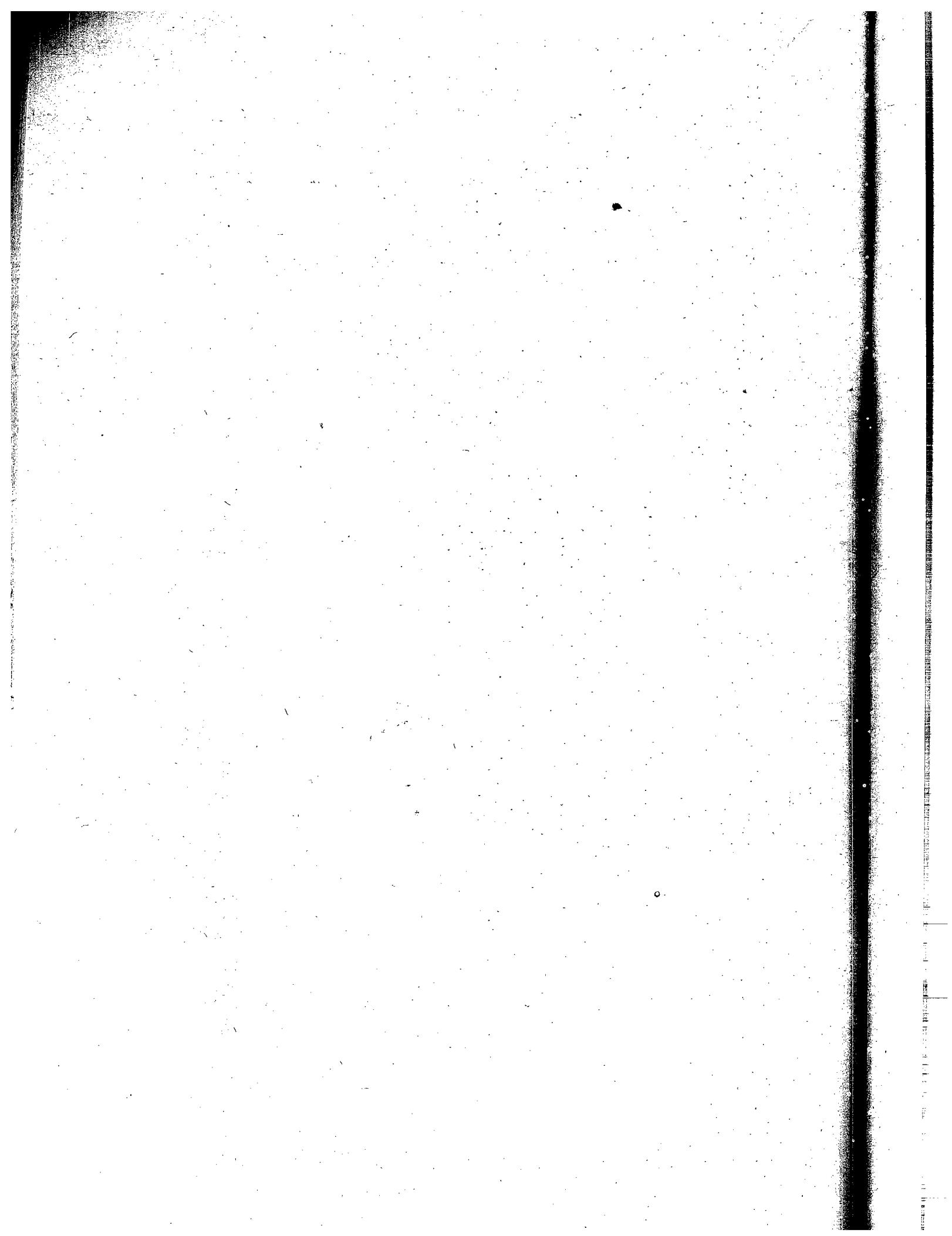
Icon () → Tomcat 6.0 X → Start.

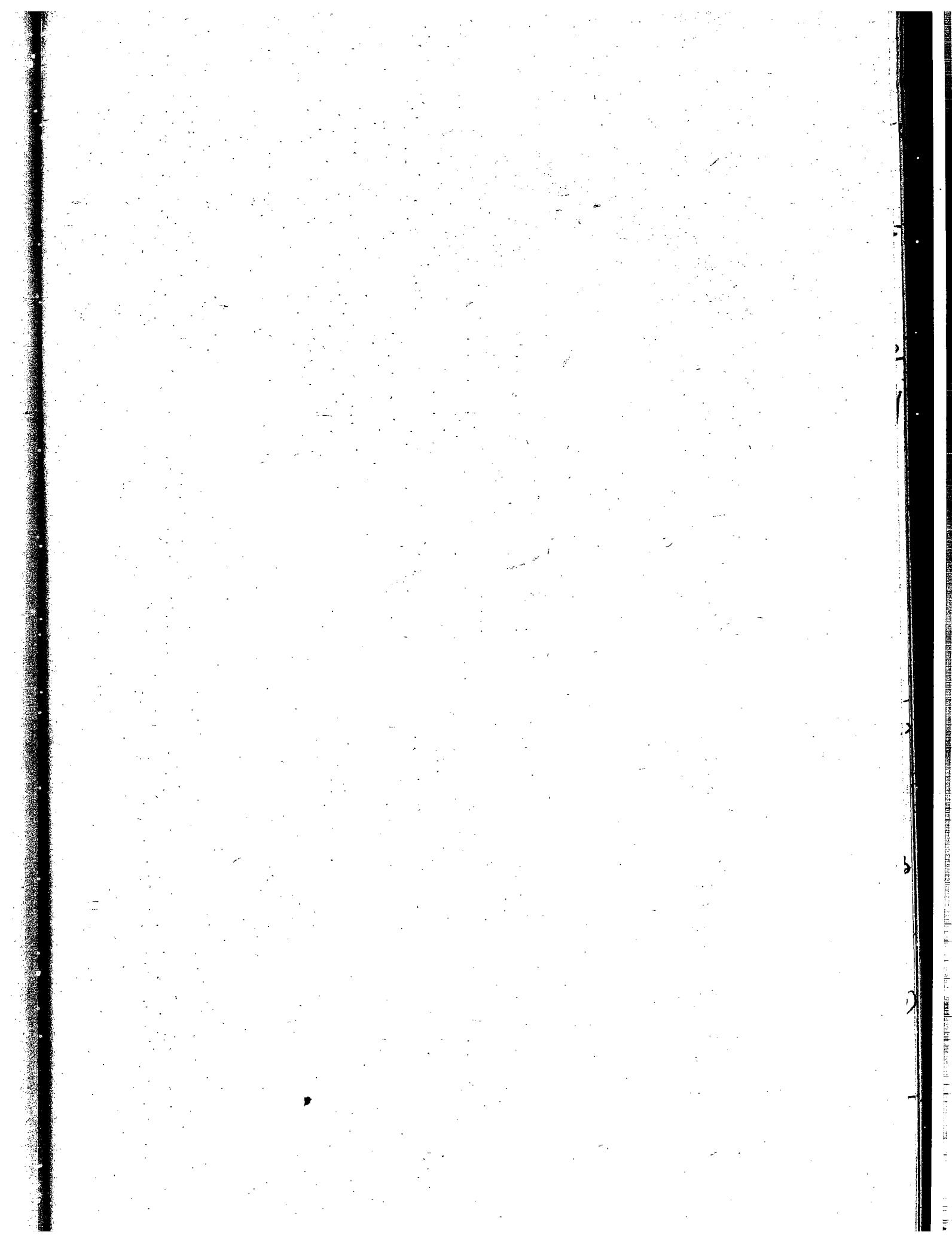
Step 10:

click on window menu → Show View →
Web Browser → Enter the following URL
at address bar (http://localhost:2011/TestApp/
login.html →

Enter.

Note:- If we do any modifications in the
servlet code then we no need to redeploy &
restart the server. The changes are automatically
effected.





(2) → Server receives the request from client. Server checks if the request is valid or not. If valid, then server transfers the request to container.

(3) → Container immediately creates a request object, response object and a thread.

(4) → Container stores request and response objects into thread object.

(5) → The thread will call service() method of servlet by passing req and res objects as parameters.

(6) → The business logic implemented in service() method is executed.

(7) → The container will take the generated response from the servlet.

(8) → The container will provide the response data to server.

(9) → Container immediately destroys/removes the req object, res object and thread object/thread.

(10) → Server will construct the response data in the form of a web page and then server will send the web page back to client browser.

Note

- For Every request, container creates a pair of request and response objects and a thread object.
- For example, if three requests are given to a servlet then three pairs of request and response objects are created and three threads are created by the container.
- All request and response objects and threads will share a single OS level process. So, there will be a less burden on the server.

Extending our servlet from GenericServlet:-

- GenericServlet is a predefined class given by the servlet API and this class is implemented from the following two interfaces.
 - i) Servlet interface
 - ii) ServletConfig interface.

→ GenericServlet class is an abstract class, because this class is containing one abstract method called service(`req, res`) method.

- Servlet interface has provided five abstract

four methods and one method is not implemented
i.e., service() method.

→ ServletConfig interface has given four abstract
methods and GenericServlet class implemented all
of its four methods.

→ so, there is only one abstract method in
the class called service.

→ if we extend our class from GenericServlet
class then we must implement service() method
in our servlet class.

→ if we do not override service() method in
our class then our class becomes an abstract
class.

→ If our servlet class is an abstract then
container doesn't create any object for our
servlet class. So we must override service() method
in our class.

public abstract class GenericServlet implements Servlet,
ServletConfig

8

interface given implemented abstract
Servlet

public class MyServlet extends GenericServlet

public void service(ServletRequest request, ServletResponse response)
throws ServletException, IOException

→ while extending our servlet class from GenericServlet,
the life cycle methods init() and destroy()
methods are optional to override but
service() method is mandatorily to override
in our class.

Q:

Why service() method is made as abstract method in
the GenericServlet class?

Ans:

Service() method is an important life cycle
method, which is executed for each
request

→ if a programmer wants to provide
business logic in a servlet then the

programmer need one method in the class.

so, the servlet API developers made service() method as abstract and informed that the servlet programmer must override this service() method to implement the business logic required.

→ if the API developers only given service() method implementation also then the servlet programmer doesn't have any methods to implement his business logic. so, they made service() method as abstract.

Printing response on to a browser(Client):-

↳ → while writing a servlet class, after processing is completed the servlet must print its response on to a client browser.

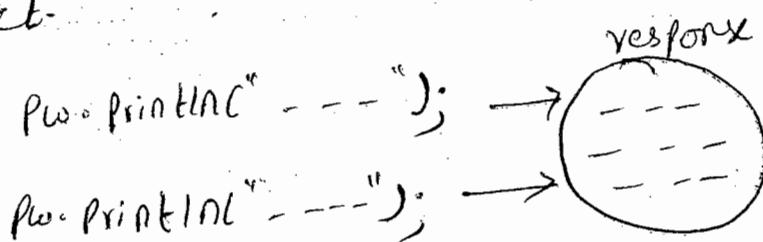
→ if we write System.out.println() statement in a servlet then it will be printed on server console, but not on to the client browser.

→ If we want to print any response on to the browser, we should store the response data into response

- In order to write the data into response object, we need a stream object called PrintWriter.
- The PrintWriter object will be given to a servlet programmer by response object only. so we need to get PrintWriter object through response object in our servlet by using the following statement.

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

- if we write any output using PrintWriter object then it will be stored into response object.

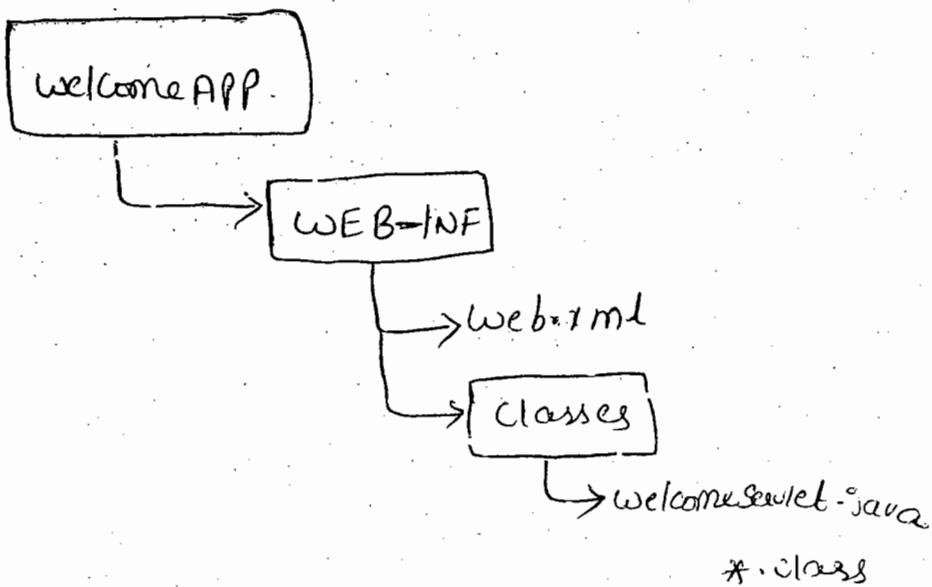


- The web container will receive response object from our servlet and provides the response data to the server.

- The server constructs a web page with the data given by the container and then the server will send that web page back to a client browser.

Example:

The following web application is for printing a welcome message along with system date and time on to the client browser.



//welcomeServlet.java

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

```
public class welcomeServlet extends GenericServlet
```

```
{  
    public void service(ServletRequest request, ServletResponse response)  
        throws ServletException, IOException
```

}

```
Date d = new Date();
```

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

```
pw.println("Date & time :" + d.toString());
```

```
pw.println("welcome to servlet");
```

```
pw.close();
```

}

}

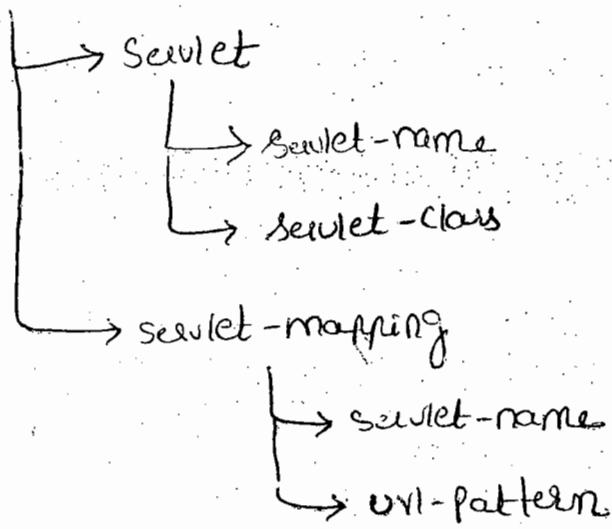
Deployment Descriptor file (web.xml):-

- For each web application, there must be one web.xml file is required.
- whenever we deploy our web application into a server then the web container first reads web.xml file immediately.
- Each servlet created by the programmer must be configured into web.xml file.
- If we do not configure a servlet in web.xml file then container doesn't create an object for that servlet class.
- while configuring a servlet in web.xml, we should provide three names for it.
 - 1) digital class name (fully qualified class name)
 - 2) logical name or alias name for dummy name

- The logical name will be used by the container as object name. It means object name will be provided by the programmer and object of the class will be created by the container.
- we call logical name also as an instance names of servlet created by the container.
- The web container identifies whether a client request is given to this servlet class or not, by using url pattern provided for the servlet in web.xml
- In order to configure a servlet in web.xml file, we need to use the following two elements (tags)
 - 1) <Servlet>
 - 2) <Servlet-mapping>
- The root element of web.xml is <web-app> and we should write the above two elements inside the root element.

web.xml

web-app



Note :

The comments in html, jsp, xme all are same

web.xml

<!-- web.xml --> --> comment

<web-app>

<Servlet>

<servlet-name> welcome </servlet-name>

<servlet-class> WelcomeServlet </servlet-class>

</Servlet>

<servlet-mapping>

<servlet-name> welcome </servlet-name>

<url-pattern> / or / </url-pattern>

</servlet-mapping>

</web-app>

compiling a servlet :-

- In case of JEE/J2EE technologies, the software is nothing but one or more jar files
- For servlet or JSP or EJB or any framework, software will be in the form of jar files.
- For Technologies like servlet, JSP and EJB, the software will be provided by server vendors.
It means we will get jar files along with a server software.
- In case of Tomcat server, servlet software is servlet-api.jar
- Before we compile our servlet, we need to set the servlet-api.jar file in the class path.
D:\welcomeAPP\WEB-INF\classes > set classpath =
C:\Tomcat6.0\lib\servlet-api.jar;%
classpath %.
- D:\welcomeAPP\WEB-INF\classes > javac WelcomeServlet.java

Deployment into Tomcat :-

- Copy the web application root directory (welcomeApp) into C:\Tomcat6.0\webapps folder (Hard deployment)

Starting the sever:-

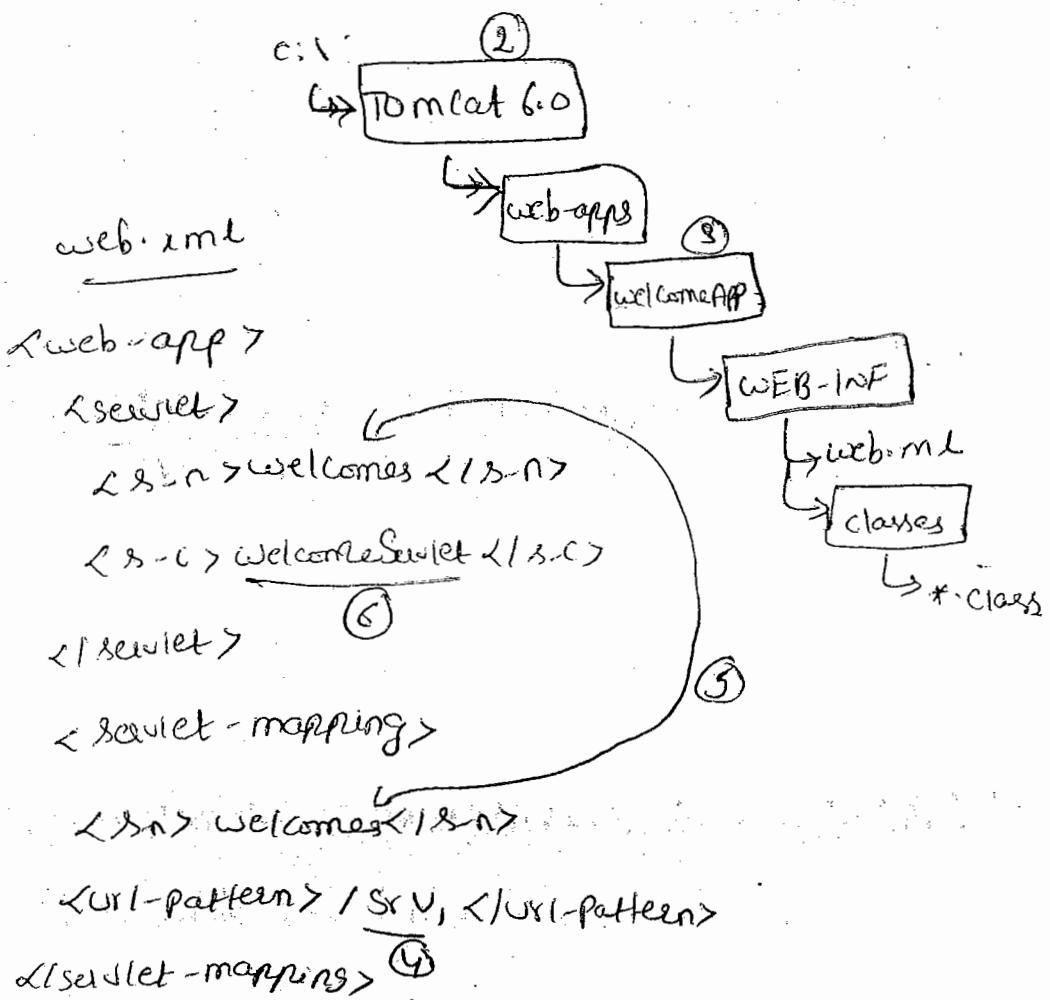
Double click on Tomcat 6.exe available at
C:\Tomcat 6.0\bin folder

sending a request to servlet:-

→ Open the browser and type the following address.

http://localhost:2011/welcomeApp/srv,

http://localhost:2011/welcomeApp/srv ①



can we put multiple url patterns to a single servlet class or not?

yes, possible - In web.xml file, we have to repeat

<servlet-mapping> tag for multiple times

→ For example, if we want to put multiple url patterns to the above servlet then in web.xml, we should add the following <servlet-mapping> elements.

<servlet-mapping>

<servlet-name> welcomes </servlet-name>

<url-pattern> /srv₂ </url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name> welcomes </servlet-name>

<url-pattern> /srv₃ </url-pattern>

</servlet-mapping>

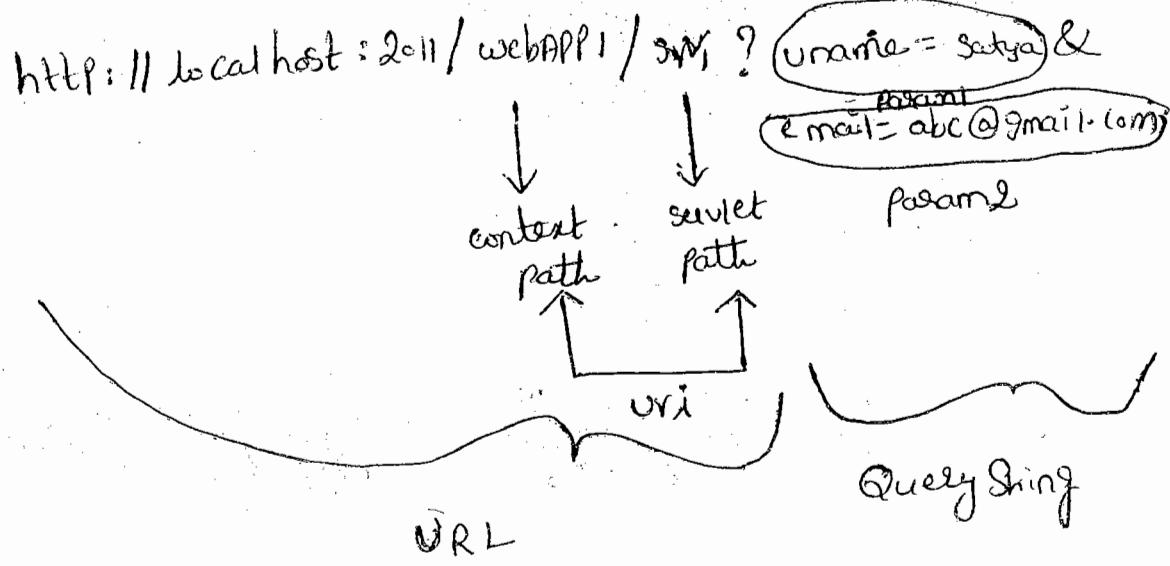
Adding parameters to a request from browser

→ while sending a request to a servlet from browser, along with request we can also

pass some parameters to the url of

the servlet

- If we want to add any parameters to the url in the address bar, we need to separate the url and the parameters by putting "?" symbol.
- If we want to add any parameters to the url in address bar then we call it is a query string.
- we can add multiple parameters also to the url in address bar. In this case, we need to separate one parameter and another parameter by putting "&" symbol.



- When we pass any parameters to a servlet from the browser, then container will store all the given parameters in request object.
- If a servlet wants to read the input value given by the client then in

service() method of ~~Servlet~~,
input values by calling getParameter() method
given by ~~Servlet Request~~ interface.

→ In service() method of a servlet, we need the
following code for receiving the values of uname
and email parameters.

public void service(ServletRequest request, ServletResponse response)

throws ServletException

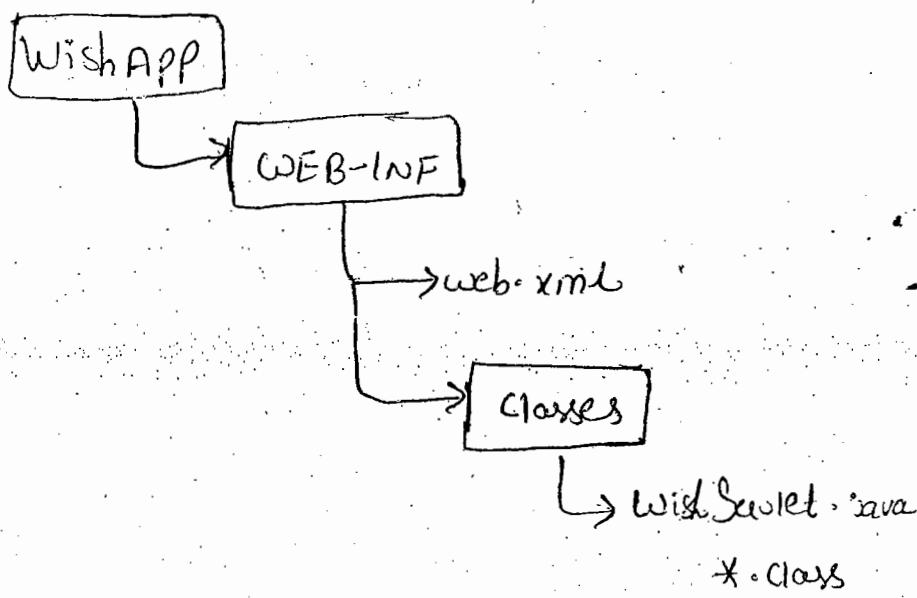
{
String s₁ = request.getParameter("uname");

String s₂ = request.getParameter("email");

Ex:-2

The following web application accepts username
parameter from the browser along with request
to servlet and prints some welcome message

for the given user name.



// WishServlet.java

```
import javax.servlet.*;
```

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

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

```
public class WishServlet extends HttpServlet
```

{

```
    public void service(ServletRequest req, ServletResponse res)
```

```
        throws ServletException, IOException
```

{

```
    String user = req.getParameter("username");
```

```
    Calender c = Calender.getInstance();
```

```
    int h = c.get(Calendar.HOUR_OF_DAY);
```

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

```
    if(h >= 6 & h <= 12)
```

{

```
        pw.println("<font color='green' size=8>");
```

```
pw.println("</font>");
```

{

```
else if(h>=13 && h<=18)
```

{

```
pw.println("<font color='blue' size=8>");
```

```
pw.println("Good afternoon:" + user);
```

```
pw.println("</font>");
```

{

else

{

```
pw.println("<font color='red' size=8>");
```

```
pw.println("Good night:" + user);
```

```
pw.println("</font>");
```

{

```
pw.close();
```

```
} // service ..
```

```
} // class ..
```

web.xml

```
<!-- web.xml -->
```

```
<web-app>
```

```
<servlet>
```

```
<servlet-name> wisher </servlet-name>
```

```
<servlet-class> WishServlet </servlet-class>
```

< servlet-mapping >

< servlet-name > wishes < servlet-name >

< url-pattern > /Srv₁ /Srv₂ < url-pattern >

< /servlet-mapping >

< web-app >

→ In the above servlet class, we imported java.util package, because of Calendar class.

→ Calendar is an abstract class, so we cannot create an object with new keyword.

→ To get a reference of Calendar class, we called a factory() method called getInstance()

→ A factory() method means it is a method of class returns an object.

→ In Java we have two types of factory methods

1) static factory() method

2) Instance factory() method.

→ compile the above servlet and deploy the web application into tomcat

→ Start the tomcat sever.

→ Open the browser and type the following

<http://localhost:2011/wishful/>

0/p:

Good night : Soviet

white ♀

ed

b

~~29/10/11~~ Tomcat Server

→

- While installing Tomcat server by default the server will be installed at a location

C:\Program Files\Apache Software Foundation\Tomcat 6.0

- While installing the server, we can change the location (for example, Tomcat 6.0)

- When we install the Tomcat server then it will be by default installed with port no 8080. If

The system has database, also then Oracle http

Server also runs on 8080 port number so there is a clash between the two port numbers.

- If we send a request with port no 8080 then the request will be given to the Oracle HTTP server, but not to the Tomcat server. So we need to change the Tomcat ^{server} port number.

- If we want to change the Tomcat server port number then we need to open server.xml file available at

C:\Tomcat 6.0\conf folder

we

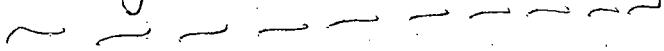
or

→ In Server.xml, select & go to <Connector> element (line no: 69) and change the port number.

<Connector port="2011" protocol="HTTP/1.1"

→ If already server is started then we need to shut-down and restart the server again.

Starting the Tomcat server



Three ways:

way 1:

Start → programs → ApacheTomcat6.0 → configure Tomcat →
(click)

General → start / stop

way 2:

Start → programs → ApacheTomcat6.0 → Monitor Tomcat →

In the system tray of task bar, right click on Tomcat
Server icon → start service

way 3:

Double click on Tomcat6.exe available at

C:\Tomcat6.0\bin folder

Note:

- The third approach is the standard approach to start the Tomcat server.
- In Tomcat server, catalina is called a servlet container and jasper is called ajsp engine.

Note:

- Apache web server is different from Tomcat web server
- The implementation classes provided by Tomcat server for the servlet API are available at catalina.jar

Servlet Request and Servlet Response:-

- In the service() method of a servlet, there are two parameters and those are references of ServletRequest and ServletResponse interfaces.
- Both ServletRequest and ServletResponse interfaces are interfaces given in "javax.servlet" package.
- When a client request is given then the container creates implementation classes objects of ServletRequest and ServletResponse interfaces. Container will pass those objects to our code and we receive into

the interface references.

→ If we directly write the class names in the service() method, instance of interface names then our servlet becomes server dependent. so we use interface names, to make our servlet as server independent.

→ If we want to print the implementation class names of SovietRequest and SovietResponse on to the browser then we need the following statements in the service() method.

```
pw.println("SovietRequest's implementation class is:" +
```

```
request.getClass().getName());
```

```
pw.println("SovietResponse's implementation class is:" +
```

```
response.getClass().getName());
```

<load-on-startup> element:-

→ By default a web container creates an object for servlet, whenever first request is given to it.

→ For first client the response will be given slow, because the client has to wait until servlet object is created and initialized.

→ For second request onwards, the response will be faster than first request. In order to eliminate this difference, we need to inform the container that, create servlet object before first request is given. For this we need to add <load-on-startup> element into the web.xml file (DD file). **

→ If we add <load-on-startup> Element then the container creates servlet object, whenever our web application is deployed & whenever the server is started.

→ <load-on-startup> Element should be included inside <Servlet> Element.

→ For example,

```
<Servlet>
  <servlet-name> welcome </servlet-name>
  <servlet-class> WelcomeServlet </servlet-class>
</servlet>
```

<load-on-startup> 1 </load-on-startup>

```
</Servlet>
```

→ <load-on-startup> behavior will be like the following:

- a) The value must be a positive integer
 - b) Range of <load-on-startup> is 0 to 65,535 (it is server dependent)
 - c) low value gets high priority
 - d) high value gets low priority
 - e) if same priority given for multiple servlets then the order will be decided by the container
- **
- f) if we give a negative value then the container ignores load on startup and creates a servlet object when first request is given. but container doesn't throw any exception.

Note:

→ up to Tomcat 5.5, if we provide load on startup value as 0 then container provided least priority and that servlet object is created after creation of other servlet object with <load-on-startup> from but since Tomcat 6.0 for <load-on-startup> value 0, the container provides highest priority to create object.

Tomcat 5.5

servlets

servlet 1

servlet 2

servlet 3

servlet

<load-on-startup>

10

5

0

Order

2

1

3

-1 (ignores)

(when request
is given)

Tomcat 6.0

Order

3

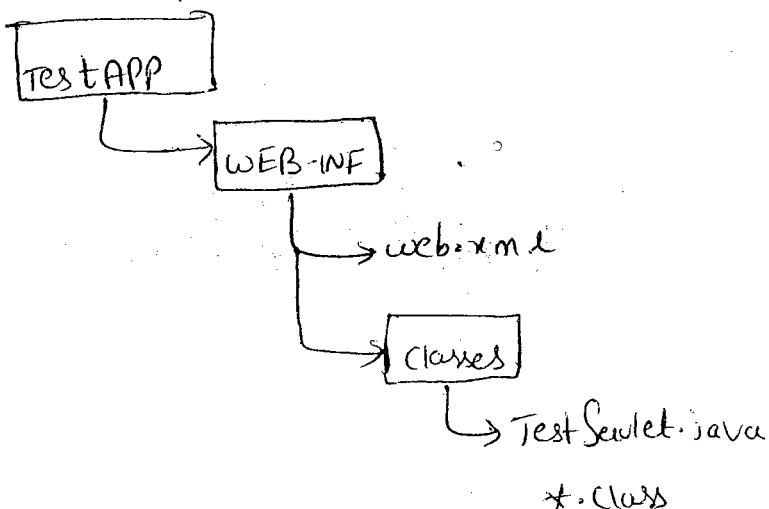
2

(when request is
given)

1

30/07/11
Ex:-3

→ The following web application is to test load on startup
of a servlet.



1/TestSoviet.java

```
import java.soviet.*;
```

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

```
public class TestSoviet extends GenericSoviet
```

```
{
```

```
    public void init(SovietConfig config) throws SovietException
```

```
{
```

```
    System.out.println("init method called...");
```

```
} // init()
```

```
    public void service(SovietRequest request, SovietResponse response)
```

```
        throws SovietException, IOException
```

```
{
```

```
    System.out.println("service method called....");
```

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

```
    pw.println("<h2>" + this + "</h2>");
```

```
    pw.close();
```

```
} // service()
```

```
    public void destroy()
```

```
{
```

```
    System.out.println("destroy method called...");
```

```
} // destroy()
```

```
}
```

web.xml:

<!-- web.xml -->

Wt

<web-app>

→

<servlet>

<servlet-name> Test </servlet-name>

→

<servlet-class> TestServlet </servlet-class>

<load-on-startup>1 </load-on-startup>

</servlet>

<servlet-mapping>

→

<servlet-name> Test </servlet-name>

<url-pattern> /testSrv </url-pattern>

</servlet-mapping>

<web-app>

→ In web.xml file, <servlet-mapping> tag is fixed -

it means we cannot add any additional

element into this tag.

→ compile the above servlet, and deploy the
web application into the sever.

→ start the sever and then observe the

sever console, in that we will get an

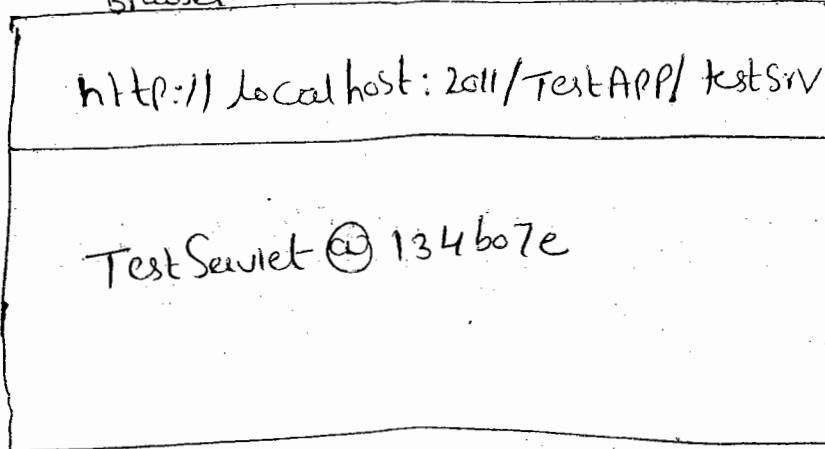
SOF message init method called, before any

request is given to the servlet because

We have added <load-on-startup> tag in web.xml.

- After sending a request, if we observe the console then we can find service() method called statement/message on the console.
- If we send multiple times request from the browser then the service() method called statement will be printed for multiple times on the console.
- Go to web apps folder of the Tomcat server and delete the web application (Test APP) then on server console we can find a statement destroy() method called.

Because:



Q:

How can we prove that only one object is created for a servlet.

Ans:

We can prove by printing the memory block address of the servlet object on to the browser/console.

→ To print the memory block address, in `service()` method of servlet we need to add the following statement.

`pw.println(this);`

Q:

Q: Why each servlet class must be public?

Ans

For a servlet, an object is created by the web container. If the container wants to create an object then the servlet class must be visible to the container.

→ To make our servlet class as visible to the container, our ^{servlet} class must be public.

Q:

How container is creating an object of a servlet?

Ans

Ans:

Container first loads the Soviet class and then calls new Instance() method for creating an object of a Soviet.

Test Soviet test = (TestSoviet) class.forName("TestSoviet").
newInstance();

Q:

Can I write an user defined method in a Soviet class or not?

Ans:

It is possible to write user defined methods in a Soviet class but container doesn't call the user defined method because user defined method is not a life cycle method.

→ user defined method can be called from life cycle methods by the programme

public class TestSoviet extends GenericSoviet

{

 public void service(SovietRequest req, SovietResponse res)

{

 System.out.println("In service method");

public void m1()

2

g

y

Q: can we write main method in a servlet class or not?

Ans: yes, because main method is not a life cycle method
so container doesn't call the method.

→ According to servlet specification, main method
is not recommended to place in a servlet
class.

Q:
can we place a parameterized constructor in a
servlet class or not?

Ans: In each servlet class, default constructor/no argument
constructor is mandatory. if we do not add
any constructor in the class then while
compiling the java class & servlet class, the
compiler automatically adds default constructor

→ To select multiple options in a list box, we need to hold control key.

→ Each option need a value and whenever we submit the form then the option values will be submitted to server (Servlet).

// BookServlet.java

```
import javax.servlet.*;
```

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

```
public class BookServlet extends GenericServlet
```

```
{
```

```
    public void service(ServletRequest req, ServletResponse res)
```

```
        throws ServletException, IOException
```

```
{
```

```
    double price=0;
```

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

```
    String books[] = req.getParameterValues("book");
```

```
    for(int i=0; i<books.length; i++)
```

```
{
```

```
    if(books[i].equals("java"))
```

```
        price = price + 250;
```

```
    if(books[i].equals("net"))
```

```
if (books[i].Equals("Oracle"))
```

```
    price = price + 150;
```

```
if (books[i].Equals("CPP"))
```

```
    price = price + 100;
```

```
}
```

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

```
pw.println("<font color='blue'>");
```

```
pw.println("username:" + user);
```

```
pw.println("<br>");
```

```
pw.println("selected books:");
```

```
for (int i = 0; i < books.length; i++)
```

```
    pw.println(books[i] + " ");
```

```
pw.println("<br>");
```

```
pw.println("</font>");
```

```
pw.close();
```

```
}
```

```
y
```

15/8/11:-

working with multiple submit buttons in a Form :-

(i) while designing HTML form to accept input value from the client, we can place multiple submit buttons for multiple operations required. and we can send a request to a servlet whenever the submit button is clicked.

(ii) Even though we place multiple submit buttons in a form, but it is possible to click on single submit button at a time.

(iii) The reason is, to a servlet, if one request is send, we need to wait for response. we call this communication as a synchronous communication.

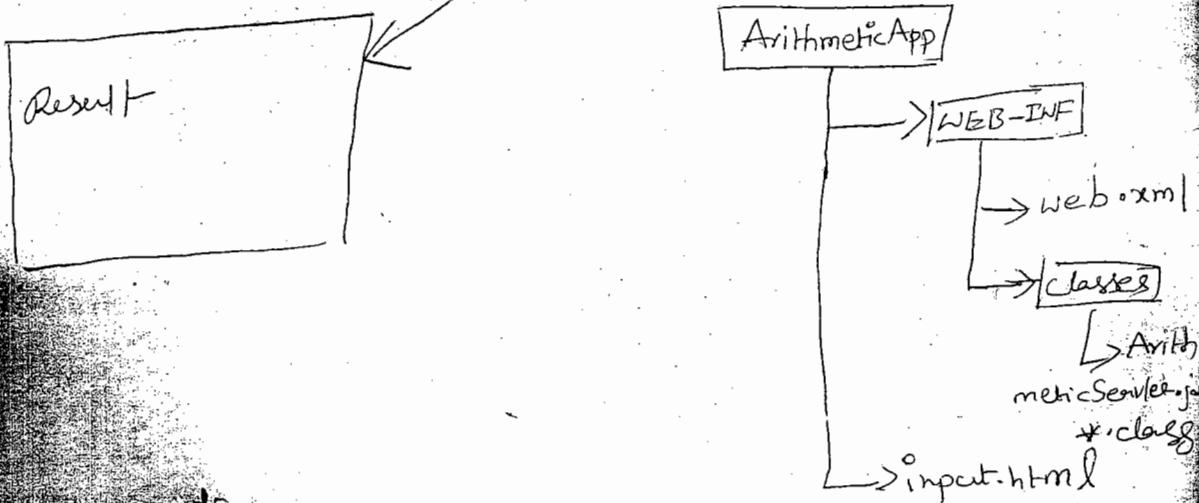
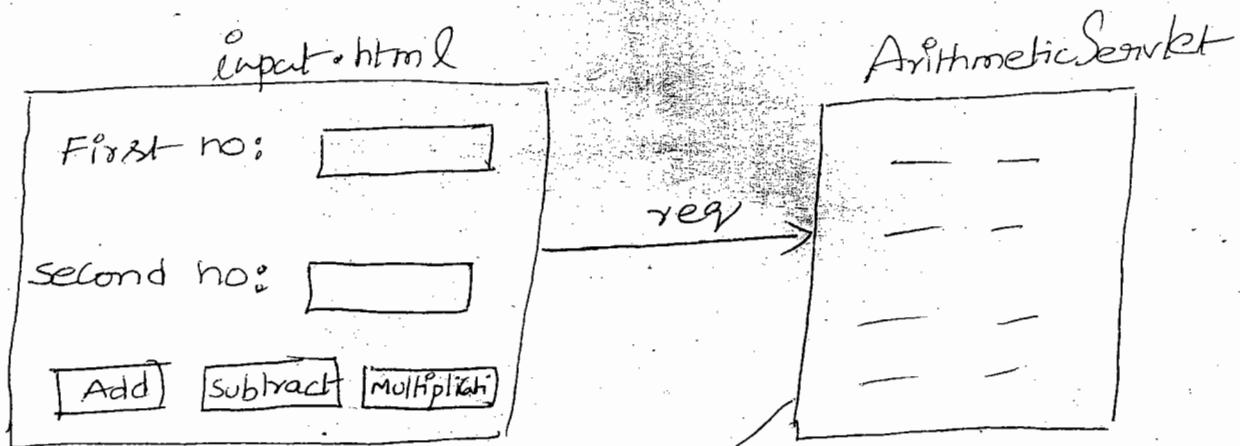
In Asynchronous communication we can send next request to a server, without getting response to the previous request. this is possible by using AJAX (Asynchronous JavaScript & XML) technology.

Send a request to the servlet
must use same name for all
submit buttons creation.

At servlet we should read the value

of the ~~submit~~ button using getParameter
submit

and through conditions we need to develop
the logic for a servlet.



<-- input type="text" -->

<Body>

<center>

<form action="Srv">

First no: <input type="text" name="tno">

Second no: <input type="text" name="sno">

<input type="submit" value="Add"

name="s1">

 (non-breakable space.)

<input type="submit" value="sub"

name="s2">

 nbsp;

<input type="submit" value="mul"

name="s3">

</form>

</center>

</Body>

ArithmeticServlet.java

ArithmeticServlet extends HttpServlet

```
{  
    public void doGet(HttpServletRequest req,  
                      HttpServletResponse res)  
        throws SE, IOException
```

```
{  
    String str1 = req.getParameter("fno");
```

```
String str2 = req.getParameter("sno");
```

```
int a = Integer.parseInt(str1.trim());
```

```
int b = Integer.parseInt(str2.trim());
```

```
String str = request.getParameter("s1")
```

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

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

```
if ("Add".equals(str))
```

```
}  
    pw.println("Addition " + (a+b))
```

```

        } else {
            pw.println("Multiplication : (" + a * b));
        }
        pw.close();
    }
}

```

web.xml

<web-app>

<Servlets>

<Servlet-name> dummy </Servlet-name>

<Servlet-class> ArithmeticServlet </Servlet-class>

</Servlet>

<Servlet-mapping>

<Servlet-name> dummy

<url-pattern>

<Servlet-name>

/srv </url-pattern>

</Servlet-mapping>

</web-app>

1) "Add".equals(str); → best one

2) str.equals("Add");

if str==null, There is a chance of getting null pointer exception. in 2

→ There is chance of getting nullpointer exception

→ Using multiple hyperlinks in a page
If we want to put multiple hyperlinks to communicate with a Servlet then we need to pass a parameter to a Servlet along with the request, the name of the parameters should be same, for all hyperlinks. When we click on hyperlink automatically, the form data is not submitted because hyperlink is not a form element.

→ We can send static values to the Servlet by adding query string.

La href = " ~~srv~~ ? fno=10 & sno=20 &
S₁ = "Add" > Add

La href = " srv, ? fno=10 & sno=20 & S₁=
Sub" > ~~Sub~~

La href = " srv, ? fno=10 & sno=20 &

S₁ = "~~Sub~~" > Mul

local

→ has

ServletConfig Interface

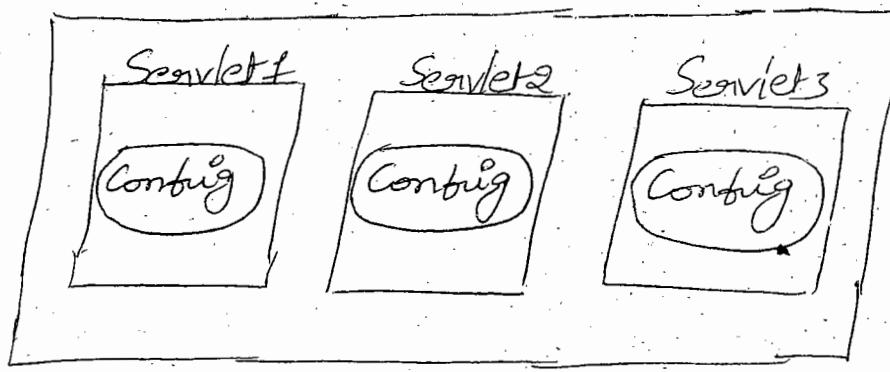
- `ServletConfig` is an interface given in `java.servlet` package and the container will create the implementation class object internally.
- The webContainer creates `ServletConfig` object, at just before calling the life-cycle `init()` method of a servlet.
- The web-Container creates `ServletConfig` object, in between execution of the constructor and the life-cycle `init()` method.
- Container first creates `ServletConfig` object and after that it creates `servlet + config` object.
- The webContainer creates a `ServletConfig` object for each servlet separately, it means `ServletConfig` is local to each servlet.
- If a web application has 3 servlets then Container

Container
Creates
Servlet object

↳ calls
`defaultConstructor`

↳ Create
ServletConfig object

↳ calls
`init (ServletConfig config)`



→ The data stored in servlet Config object of the one servlet can not be accessed by another servlet because servletConfig object local object.

⇒ The webContainer will store the following three types of information into servletConfig object.

- (i) logical name of servlet
- (ii) init parameters of servlet
- (iii) ServletContext reference.

→ A webContainer will collect logical name of servlet, init parameters of servlet from web.xml file.

→ When we deploy our application into a server & when the servlet is started then the container creates ~~Config~~ servletConfig object immediately, whenever Config object is created then container adds its reference

1) logical name of the servlet

2) init parameters of servlet

3) ServletContext interface.

* what is the diff b/w constructor & init() method of a servlet?

A:- In constructor of a servlet we can not use ServletConfig object, because ServletConfig object is created by the container after calling the constructor.

→ In the init() method of servlet we can access the data stored in a ServletConfig object.

→ init parameters are passed to a servlet from web.xml file, by adding <init-param> under tag <Servlet-Tag>.

→ one parameter will be in the form of name/value pair & we can pass any number of init parameters to the servlet from the web.xml file.

- Generally we use init parameters in web.xml to pass technical values to a servlet.
- For example, we can pass Jdbc connection properties from web.xml to a servlet.
- The advantage of passing init parameters from web.xml file to a servlet is, it is possible to modify the values easily & change the values easily by without doing the modifications at servlet, because if we modify the servlet code, then we need to recompile, reload & restart the server.
- For example if we pass driver properties from XML file, then it will be easy to change from one driver to another driver even from one database to another ~~or~~ or database.
- If we make any changes in XML file then no need of recompiling, redeploying, restarting of web application.

web.xml

<web-app>

<servlet>

<s-n> Test </s-n>

<s-c> TestServlet </s-c>

<init-param>

<param-name> driver </param-name>

<param-value> sun.jdbc.odbc.JdbcOdbcDriver

<init-param>

<param-value>

<i-p>

<p-n> url </p-n>

<p-v> jdbc;odbc;oradbsn </p-v>

</i-p>

<i-p>

<p-n> user </p-n>

<p-v> scott </p-v>

</i-p>

<i-p>

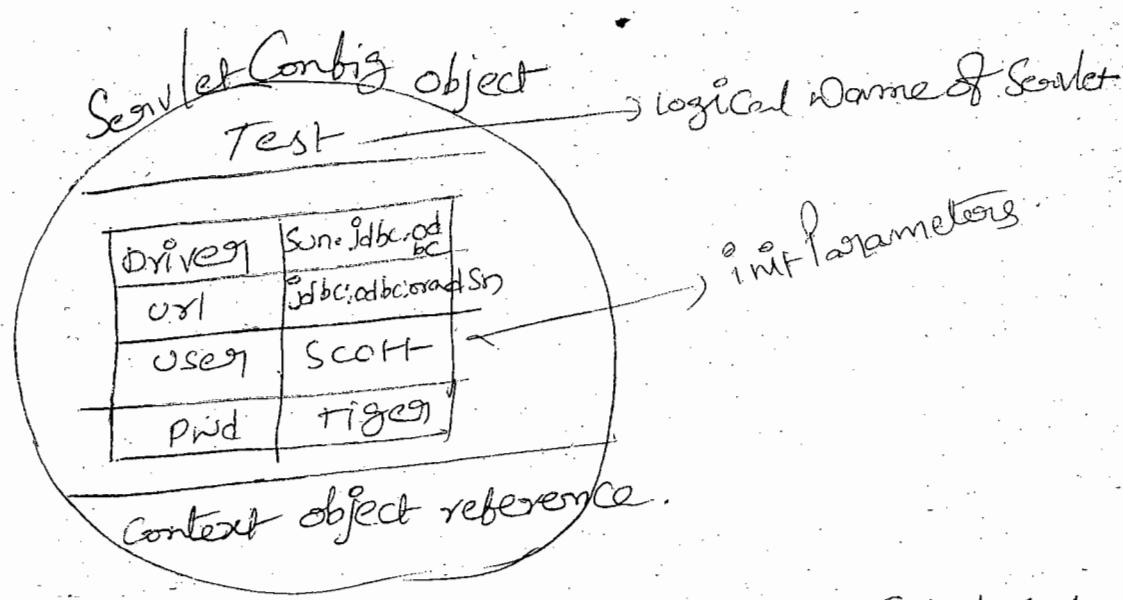
<p-n> pwd </p-n>

<p-v> tiger </p-v>

</servlets>

<s-n> Test </s-n>

→ For the above Servlet, the web container stores the following information into Servlet-Config object.



⇒ Different ways of obtaining ServletConfig

object into Service method?

→ A webContainer will pass ServletConfig to Servlet.

object to init() method of config object

⇒ But we use the data of config object mostly in service method.

→ In a Servlet we implement most of the business logic into service method.

→ So we need Servlet Config object also sometimes into the Service method.

It is prime responsibility to get Servlet-Config object into Service method and

→ As a programmer, we can obtain ServletConfig object into ~~service()~~ service() method by taking any one the following 3 approaches.

1. Approach

- In this approach, we override life-cycle init() method in our Servlet.
- Container will pass ServletConfig object by calling init() method of our Servlet class.
- Our Servlet class receives ServletConfig object into local variable by using parameter of init() method, we get the config object.
- To use ServletConfig object in service() method we need to store local ServletConfig variable into global ServletConfig variable so that we can use that global object into service() method.

For example:

P C TestServlet extends GIS

{

private ServletConfig config;

P V init(ServletConfig config) throws SE

P v service() throws SE, IOE

I use getConfig obj here.

}

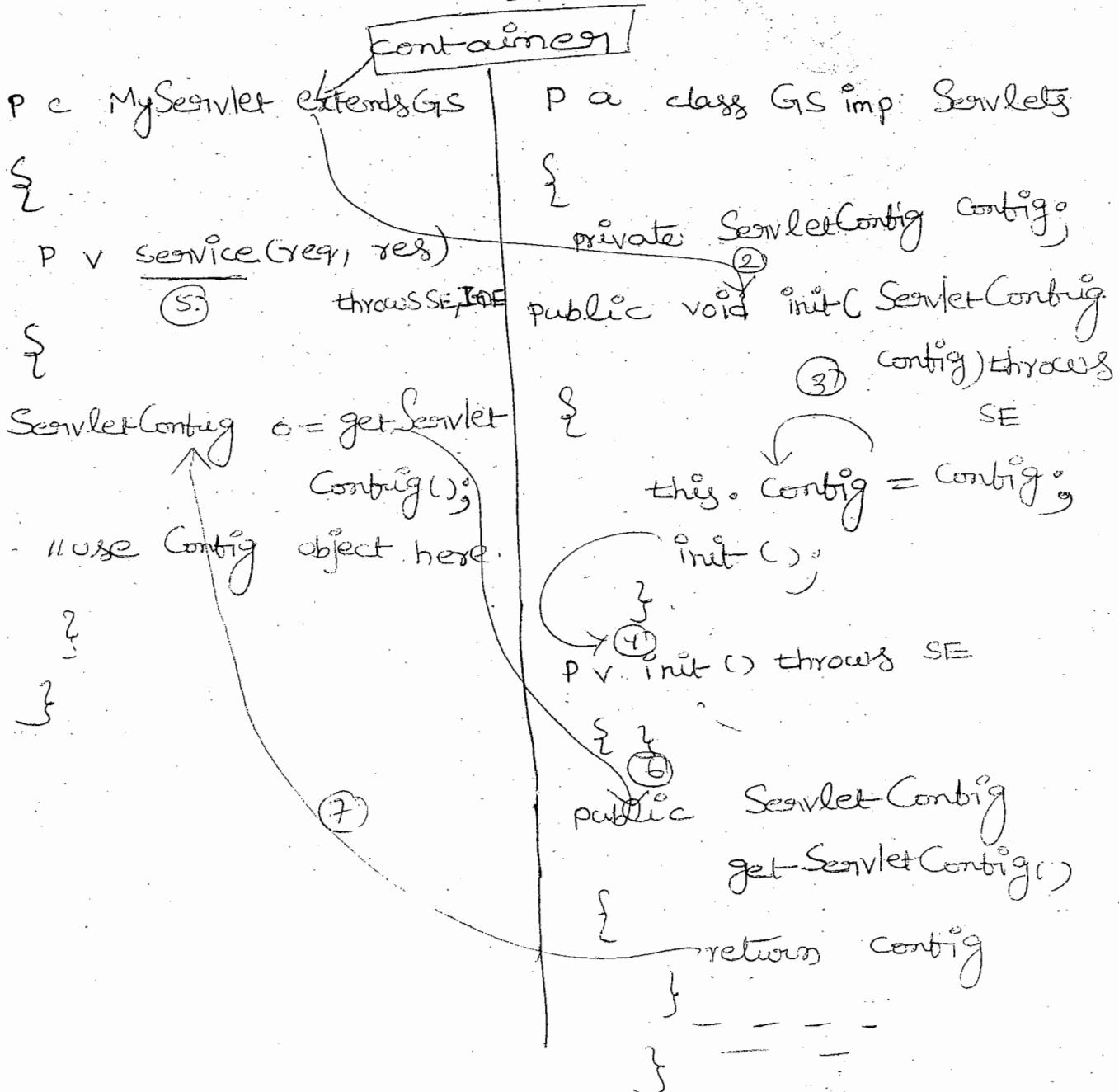
~~Date: 17/8/11~~

Q) Approach-II

- In this approach we do not override init()
- In lifecycle init() in our class
- If we don't override init() in our class then container calls base class init() by passing ServletConfig object as a parameter.
- The base class init() will store the local servlet object into a global ServletConfig object and it is of private type.
- We can't use the private variable of the base class in the derived class directly. So we can't use config object directly in our service() method.

Called `getServletConfig()` and we call that method

to get `ServletConfig` object into our `Service()`.



⇒ In Java, we can call a method directly

by without using classname (or) object name,
provided it both methods are available in same
class (or) if method is available in the base
class.

In the above approach, we are calling `getServletConfig()` in our Servlet class directly because it is provided by the base class.

⇒ GenericServlet class implemented from Servlet interface. So in GenericServlet, there are five methods which are given by Servlet interface are available.

⇒ GenericServlet class implemented four methods given by Servlet interface, but one method is not implemented i.e., `service()`. so GenericServlet is an abstract class.

⇒ Servlet interface has provided the following five methods

(i) `init(ServletConfig)`

(ii) `service(Request, Response)`

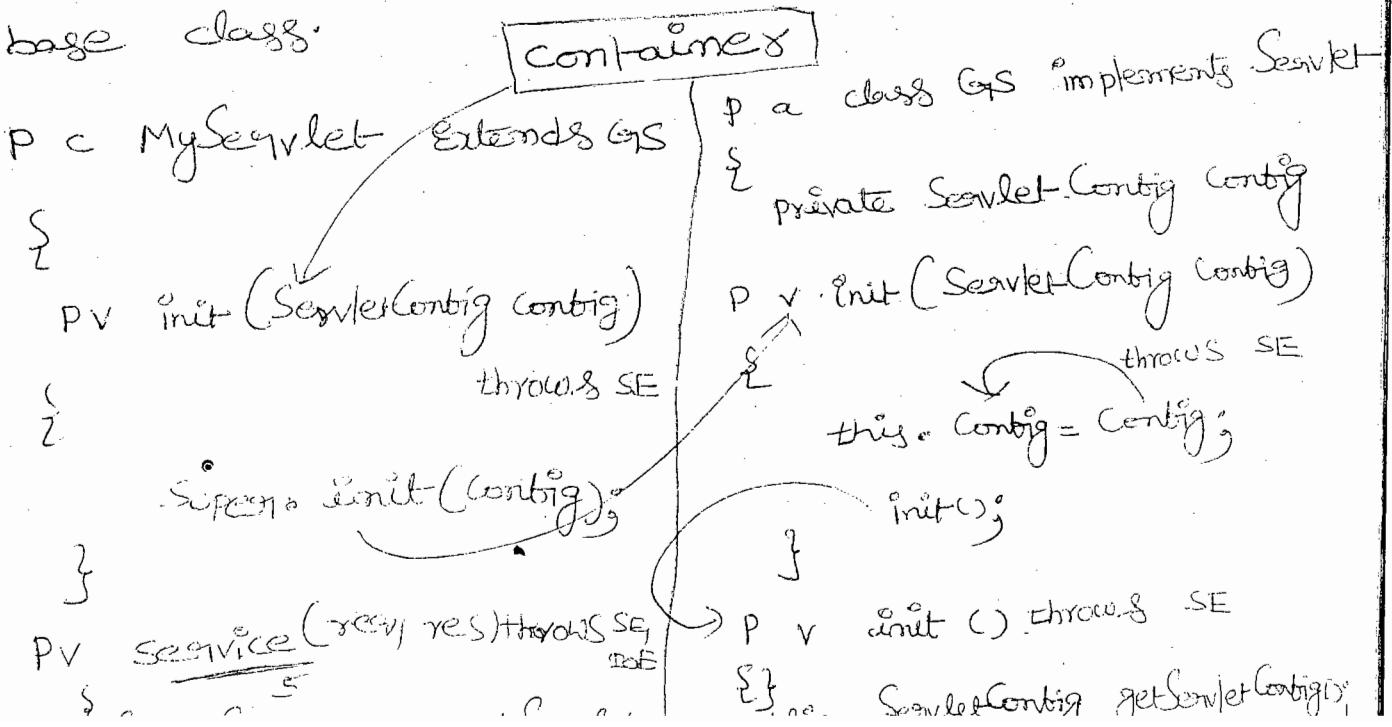
(iii) `destroy()`

(iv) `getServletConfig()`

(v) `getServletInfo()`

↳ Approach 2

- In this approach, we override lifecycle `init()` in our class.
- If we override lifecycle `init()` in our class then Container calls our class `init()` methods only.
- Because of Thread Safety reasons, we are not interested to store local variable into global variable.
- In this case we need to call base class `init()` directly from our class by using `super.init(Config)`.
- In the `service()` we get the ServletConfig object, by calling `getServletConfig()` & the base class.



Methods to read data from ServletConfig object

1) getServletName()

- This method is used for reading logical name of a servlet from ServletConfig object.
- This method returning String.

String str = Config.getServletName();

2) getInitParameter()

- This method is used to read the value of init parameter stored in the Config object.

⇒ while reading the init parameter, we will get value in String type and if required then we need wrapped into the primitive types.

String v₁ = Config.getInitParameter("P1");

3) getInitParameterNames()

- This Method is used to read all init parameter names into a "legacy collection". It is used, when there are large no of init parameters exists.

→ my method returns ---
reference and we need a loop to read the
collection

Enumeration e = config.getInitParameterNames();
while (e.hasMoreElements())
{
 Object o = e.nextElement();
 String k = (String)o;
 String v = config.getInitParameter("k");

}

4) getServletContext():

→ This method is used to get context reference
 stored in the config object.

ServletContext ctx = config.getServletContext();

~~10/2011~~ Servlet Context Interface :-

→ Servlet Context is an interface given in Java.

Servlet *; and Container creates its implementation class object internally, which is provided by the server.

→ when we deploy our web app in a server, which is already started then immediately the web container reads web.xml file and creates the ServletContext object immediately.

the ServletContext object is created when we deploy our web application in a server.

→ If we deploy our web application in a server which is not yet started then whenever the server is started, then immediately the web container reads web.xml file from the web-application and creates ServletContext objects.

Creates immediately ServletContext object.

→ The web container creates one ServletContext object per each web app.

→ For a web app, the first object created by the container is ServletContext object only.

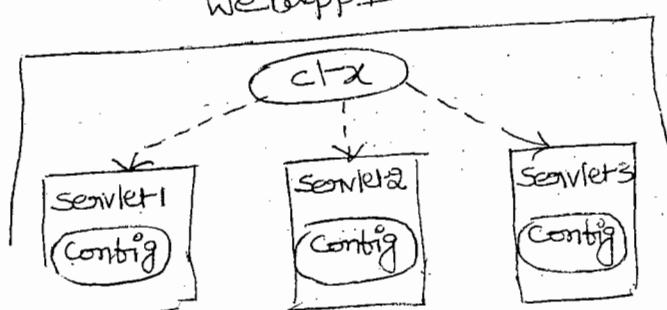
→ For a web app, the first object created by the container is ServletContext object only.

→ We call ServletContext object has a global object for a web application.

In a web app, each servlet has (S) containing

its own ServletConfig object, but all servlets will share a single ServletContext object.

webapp



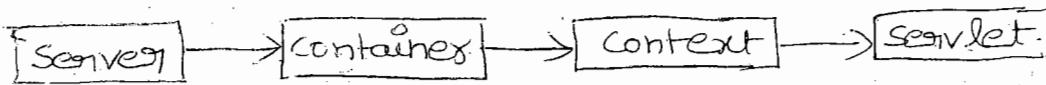
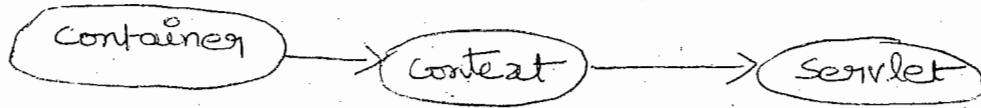
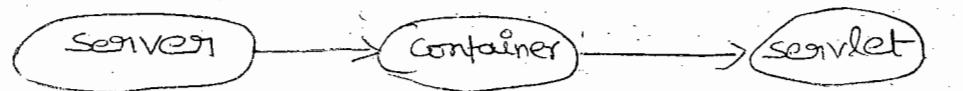
objects

⇒ Through ServletContext object, we can share some data across multiple Servlets of a webapplication.

2) Through ServletContext object, we can get the services which are provided by the container into a Servlet.

⇒ Servlet Context object acts as mediator between a webContainer and a Servlet.

⇒ A webContainer will acts as mediator between a Server and a Servlet.



⇒ The following is the sequence of the objects created by a container of a web-application.

1) Servlet Context object.

2) Servlet object

3) ServletConfig object

4) Request and Response object

5) Thread object.

⇒ Even though a web-appn doesn't contain any Servlet and JSPs, but still ServletContext object will be created by the Container.

→ To create ServletContext object, a web-container

but not for Servlets (or) JSP's.

IAQ
= Into ServletContext object, the data can be stored

in the form of key/value pairs for sharability across multiple files.

⇒ while storing the data in the form of key/value pairs into a ServletContext object, we have the following two approaches.

1) Declarative approach :-

→ In this approach, the context parameters are added into web.xml file.

→ Context parameters are global to all Servlets so we include context parameters at outside of Servlet Configurations.

→ To pass context parameters, we need an element called <Context-param>

web.xml

<web-app>

<context-param>

<param-name> PI </param-name>

<param-value> 100 </param-value>

</context-param>

<Servlet>

<Servlet

file them at the time of creating Servlet Context object by the Container, the web-container will store the context parameters into the context object.

→ From web.xml, it is possible to add two types of parameters,

(i) Context Parameters

(ii) init Parameters

Qs what is diff b/w context parameters and init parameters?

A:- Context Parameters are passed at outside of the servlet tag and this are stored in Context-Object. but init parameters are passed at inside of Servlet tag and this are stored in Servlet Config objects.

⇒ context Parameters are global to all Servlets and init parameters are local to each Servlets.
Q) How many types of parameters exists in Servlet App?

A:- Three types

(i) Request Parameters

(ii) Context "

(iii) init "

Programmatic approach

→ In this approach

the values are added (i) stored from a Servlet or a JSP.

- If we store the data from a servlet or jsp then we call it as "u can add attribute".
- we can add attribute into a context object, to share the information from the one Servlet on a jsp to other Servlets (or) a jsp available in the webapplication.

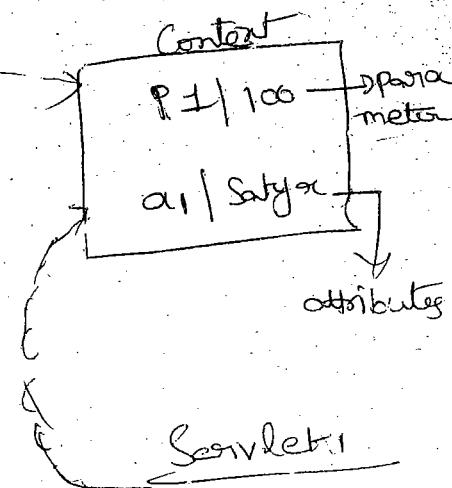
web.xml

<context-param>

<param-name> P₁ </param-name>

<param-value> 100 </param-value>

</context-param>



ctx.setAttribute

("a1", "Satya")

⇒ parameters can not modify the servlet.

parameters are

Thread Safe (only one thread act at a time).

⇒ attributes can modify the Servlets - Attribute are not Thread safe.

19/08/11

Q) what is the difference b/w a parameters and
a Attribute?

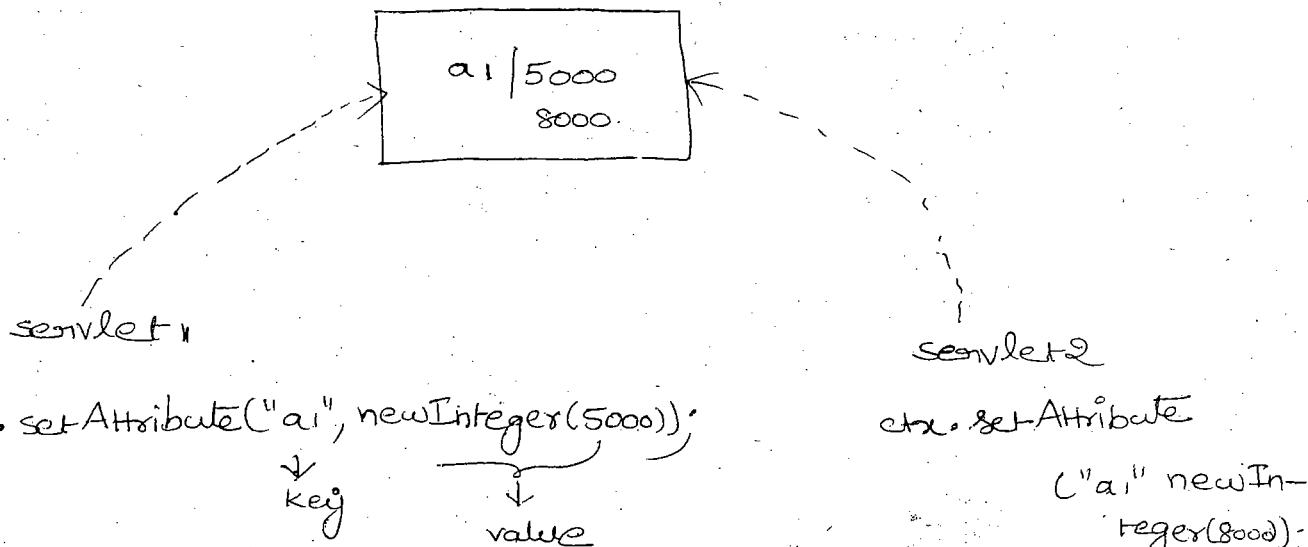
Parameters

- parameters are passed from web.xml file declaratively.
- In parameter, both key & value are of String type.
- A parameters cannot be modify from a servlet. It means parameters are thread safe.

Attribute

- Attributes are passed from a servlet programmatically.
- In an Attribute, key is a String and value is an object.
- An attribute can be modified from a servlet so attributes are not thread safe.

Ex:-



→ If we want to store, read & remove them we have the following four methods.

- 1) setAttribute(K, V);
- 2) getAttribute(K);
- 3) removeAttribute(K);

⇒ while storing
value must be an object.

⇒ while reading the value of an attribute by using its key then the value will be returned in the form of an object.

⇒ For example Object o = ctx.setAttribute("a", b);

⇒ While reading multiple attribute at once, we use getAttributeNames() and it returning enumeration

⇒ For example

Enumeration e = ctx.getAttributeNames();

⇒ In servlet API, there are 3 objects which are supporting attributes

- (i) request object.
- (ii) session object.
- (iii) context object.

⇒ Difference b/w ServletConfig and ServletContext

ServletConfig

1) Config object is a local object to a servlet.

2) Config object doesn't exist without a servletObject.

3) Config object doesn't store attributes.

Servlet-Context

1) Context object is a global object for all servlets.

2) Context object exists without a servletObject.

3) Context object stores attributes also.

Context object we have

Note. If a new ^V web application is an empty webapplication, then also ServletContext object will be created by the container.

Obtaining ServletContext object into Service()

method :-

1) `ServletContext ctx = config.getServletContext();`

2) `ServletContext ctx = getServletContext();`

⇒ A webContainer doesn't pass ServletContext object directly to the servlet. It indirectly pass context object through Config object.

⇒ When init(life cycle) is called by the Container config object will be passed as a parameter and it internally contains context object reference.

⇒ If we override init() in our Servlet class and if you store local variable into a global config then we can get context object into service() by using the "first" Syntax.

⇒ If we don't override init() (S) If we call back class init() from our class init() then we use second syntax to get ServletContext object.

Date 20/08/11 → In our Servlet class, if we override init() and if we store the local servletConfig variable into the global servletConfig variable then we follow syntax to get servletConfig into Service() method.

→ for example

P c MyServlets Extends GTS

{

private ServletConfig config;

P v init (ServletConfig config) throws SE

{

this.config = config;

}

P v service (request, response) throws SE,

IOE

{

ServletContext ctx = config.getServletContext();

}

}

not lifecycle init()

⇒ If we are overriding lifecycle init()
we are not overriding servlet config
as it we store the local servlet config
variable into the global servletConfig variable
then we follow syntax to get ServletConfig
into service() method.

Ex:-

P c MyServlets Extends GTS

{

are calling base class init() from our class

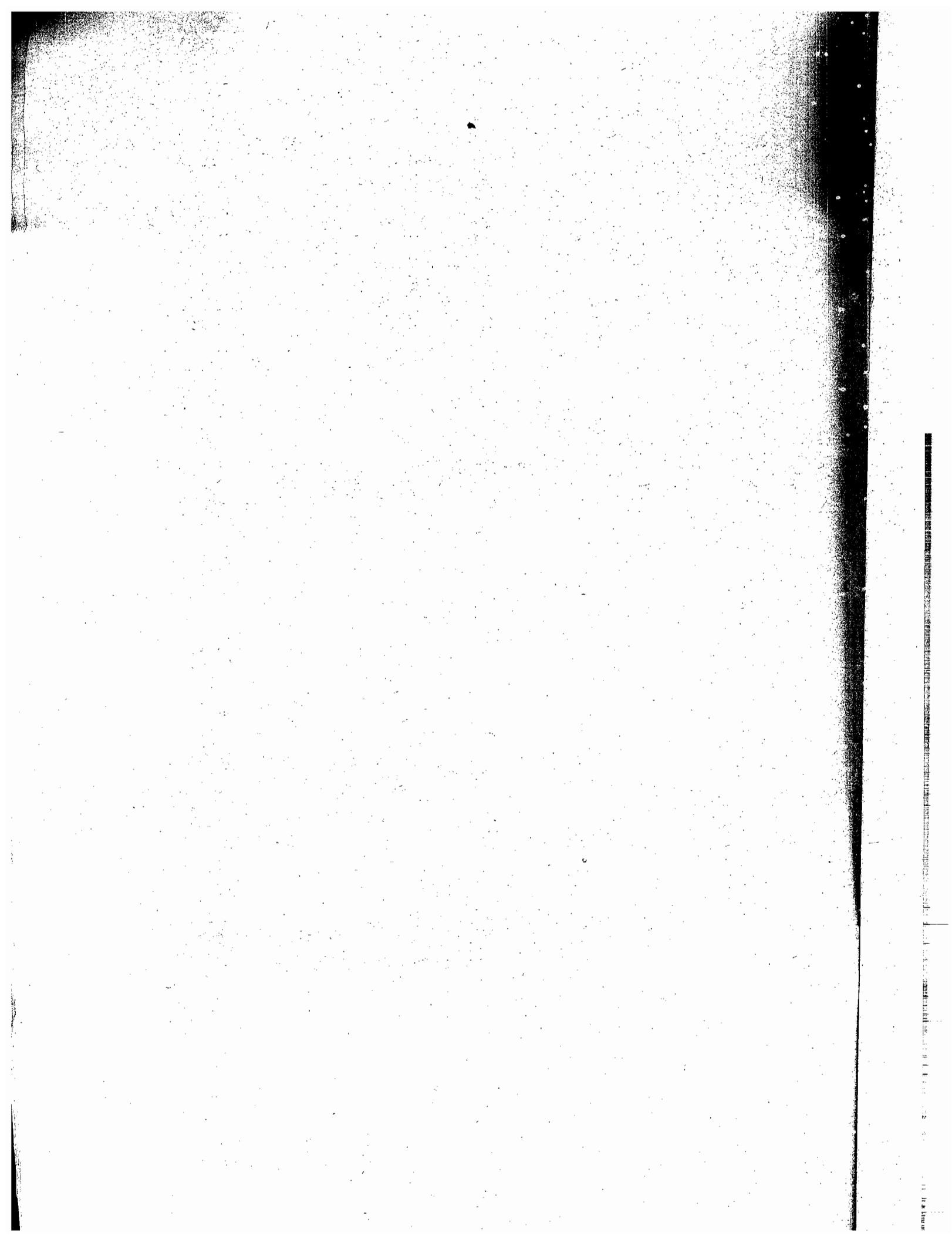
init() then we can use and syntax for getting

public class MyServlet Extends GTS

{

0
g

S
ng



```
import java.io.*;  
import javax.servlet.*;  
import javax.servlet.http.*;  
import java.util.*;  
import org.apache.commons.beanutils.*;
```

public class NewUserServlet extends HttpServlet

{

list l;

public void init(ServletConfig sc) throws

{

SE

l = new ArrayList();

super.init(sc);

}

public void doGet(HttpServletRequest req,
 HttpServletResponse res)

{

throws SE, IOException

try

{

// read input and store in a map object

Map m = req.getParameterMap();

// Create pojo - class object

UserBean ob = new UserBean();

// copy input from map object into pojo obj

" user pojo object into list

↳ add (ab)

ServletContext ctx = getServletContext();

II store list object into context

ctx.setAttribute ("users", l);

PrintWriter pw = res.getWriter();

pw.println ("new user added");

pw.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

=> In the above servlet we have created
ArrayList object for once by using init()
method because we need an ArrayList
for once in order to store all pojo class
objects.

=> In the above servlet we have improved
a new package called org.apache.commons
beandefinition. Because

object into a pojo class object, directly by calling a static method of BeanUtils class called populate()

→ In order to copy the data from map object into pojo class object, the keys of map object and variable names of pojo class, both must be same.

→ the pojo class object is added to Map and finally ArrayList object is stored in SessionContext in the form of an attribute.

1) UserBean.java (POJO class)

```
public class UserBean
{
    private String uname;
    private String email;
    private String[] cl;

    public void setUname(String uname)
    {
        this.uname = uname;
    }

    public String getUname()
    {
        return uname;
    }

    public void setEmail(String email)
    {
```

```
public String getEmail()
{
    return email;
}

public void setC1(String c1)
{
    this.c1 = c1;
}

public String getc1()
{
    return c1;
}
```

ShowUsersServlet.java

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

public class ShowUserServlet extends HttpServlet
{
    public void service(ServletRequest req, ServletResponse res) throws SE, IOE
    {
        ServletContext ctx = getServletContext();
        Object o = ctx.getAttribute("users");
    }
}
```

```
pw.println("<bs>");
```

```
pw.println("<Table border=2>");
```

```
Iterator it = f.iterator();
```

```
while(it.hasNext())
```

```
{
```

```
Object o = it.next();
```

```
UserBean ub = (UserBean)o;
```

```
pw.println("<tr>");
```

```
pw.println("<td>" + ub.getUserName() + "</td>");
```

```
pw.println("<td>" + ub.getEmail() + "</td>");
```

```
StringBuffer sb = new StringBuffer();
```

```
String h[] = ub.getCl();
```

```
for(int i=0; i < h.length; i++)
```

```
{
```

```
sb.append(h[i]); // Append method is available  
in StringBuffer.
```

```
sb.append(" ");
```

```
}
```

```
pw.println("<td>" + sb.toString() + "</td>");
```

```
pw.println("</tr>");
```

```
}
```

```
pw.println("</table></bs>");
```

```
pw.close();
```

~~set classpath = c:\root\folders\WEB-INF\lib~~

→ commons-logging 1.0.4 paste into lib folder of it
located tomcat lib.

Date

23/8/11

→ while compiling NewUserServlet.java, we will get errors, even though we have set Servlet-api.jar in the classpath.

→ because we have imported a package which is given by apache.

→ So, apart from Servlet-api.jar file, we need to set two additional jar files in the classpath.

1. Commons-beansutils.jar

2. " - collections.jar

→ the above two jar files can be downloaded separately.

→ we need to copy the above two jar files and also Commons-logging.jar into lib folder of our web-application.

→ Adding jar-files to classpath, is b7
compiling our Servlets and adding jar files into lib folder is b7 Execution of

Weblogic

→ Hard Deployment :-

- (i) web-logic is an application server, which provides domains for deploying our web-applications into the server.
- (ii) When we install web-logic server, we will get a by-default domain to deploy our applications WI-server.
- (iii) Apart from the default-domain we can also create a new-domain for deploying our applications into the server.
- (iv) Hard-deployment into web-logic server is nothing but copying our web-application root directory into the autodeploy (staging) folder of the domain.
- (v) To hard-deploy our web-application into weblogic, we need to copy the root directory into c:\bea\weblogic90\samples\domains\wl-server\autodeploy folder
- (vi) Start the weblogic server

Start → programs → BEA products → weblogic90 →

open the Browser & Type the following Request.

http://localhost:7001/BookApp/book.htm

oracle http Server Run 8080 port so we change the tomcat server port

(ii) Console Deployment :-

→ Create a war file for the web-application.

C:\BookApp>jar cvf booktest.war *

→ Start the server.

→ open the Browser & type the following url in addressbar

http://localhost:7001/console

The image shows a login form with the following fields:

- Username: weblogic
- psw: weblogic
- Login button

→ At left side Select deployments → click on

Lock&Edit button → install → upload file (click)

→
(burst)
one → select war file (BookTest.war) →

→ select war file → →

→ → → →

→ Deployment → BookTest →

~~11-81~~ → Logon

→ open the browser and type the following request in the addressbar.

http://localhost:7001/warfile name(bookTest/
book.htm)

* Creating A User - Domain in web-logic

→ By creating Domains in the server, the web-logic server maintenance becomes easy.

→ if the multiple domains are created then we can deploy applications into domains so even though we have same application name but we won't get any loss of our applications, because the applications can be deployed into different domains.

→ To create a domain the following steps are required.

Step 1:

Start → Programs → BEA products → Tools → Configuration Wizard → Next → Next → Enter user-name & psw → Next → Next → Next → Enter domain Name → Create → Done.

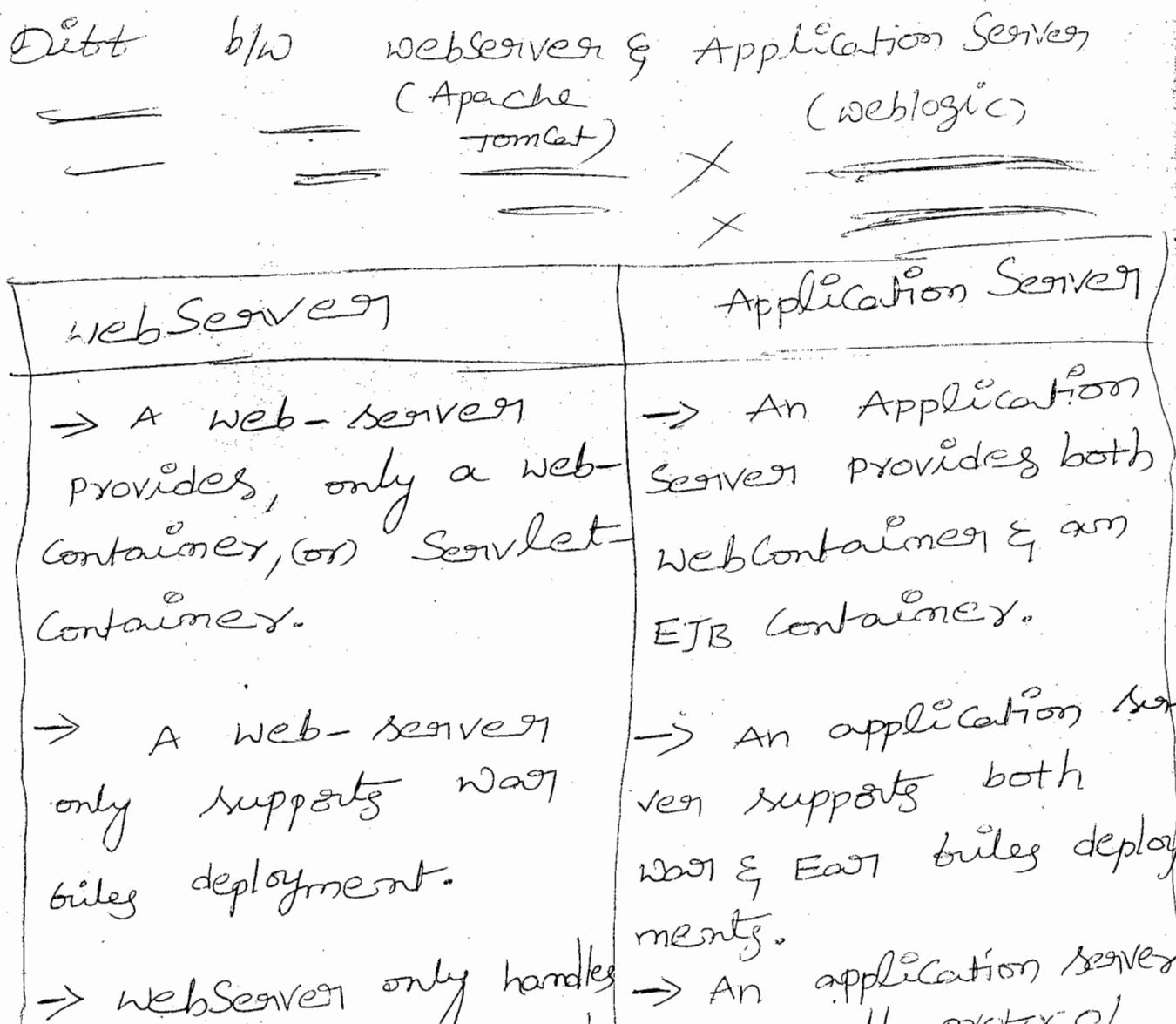
To deploy our web-application onto the domain we need to copy the root folder onto

C:\bea\user-Projects\Domains\domainName(bookTest)\autodeploy.

→ Start the Survey

Start → programming → bear → user projects

→ web-domain → startadmin Server.



less amount of middle-ware services.

→ webserver doesn't support clustering.

provided more amount of middle-ware services.

→ Application server supports clustering.

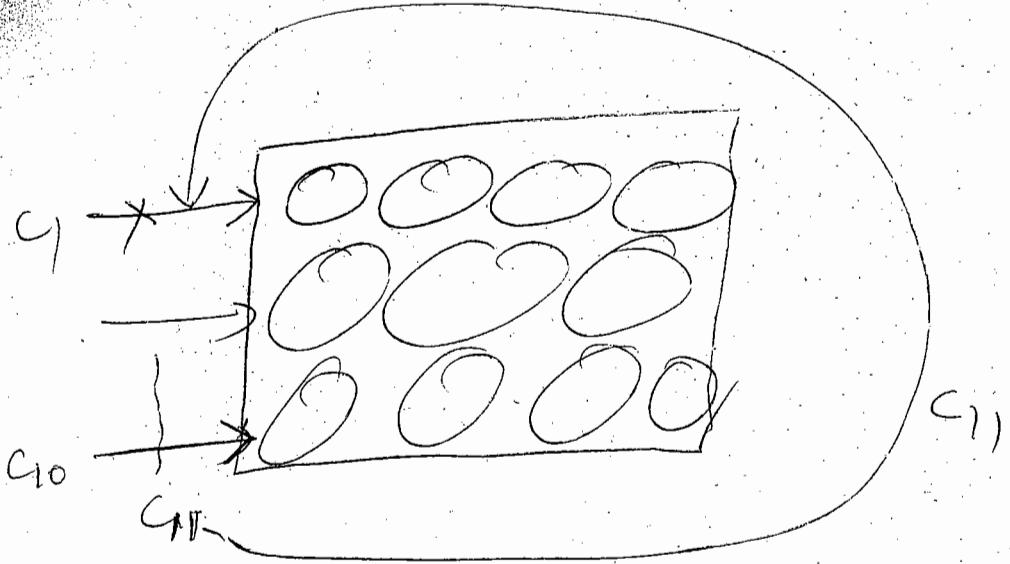
→ Note: i) An application server is also called as a web-application server.

→ An application server also contains a web container.

ii) we can not run EJB applications in a tomcat-server, because tomcat-server doesn't contain an EJB Container.

iii) Middle-ware services (or) The Real-time services required by the projects & those are like pooling, cashing, transactions, security etc.

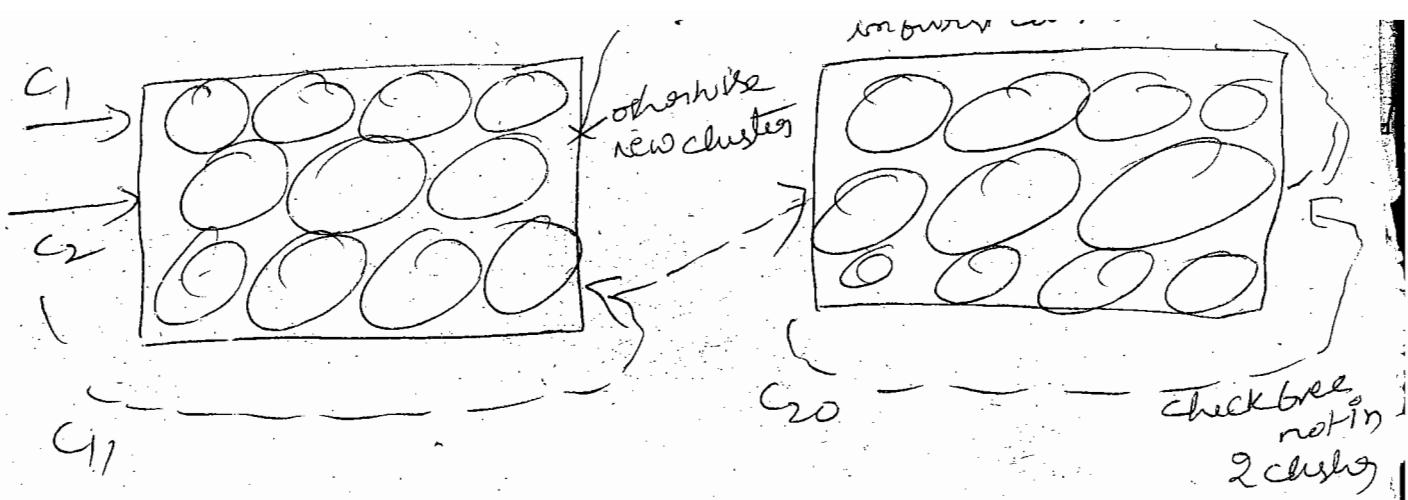
iv) A webserver doesn't support clustering, it means, when server capacity exceeds, it automatically dig-



⇒ In case of an application server it follows cluster model.

→ when that server instant capacity exceeds than automatically another instance of the server will be started & client request will be redirected to new instance.

→ If the new instant capacity also exceeds by the client, then an application server first verifies whether any free-space is available on the instances or not, if not new instance of the server is created onto the cluster.



* Serlet to Database Communication:-

→ we have the following 3 ways to communicate from a serlet to a database.

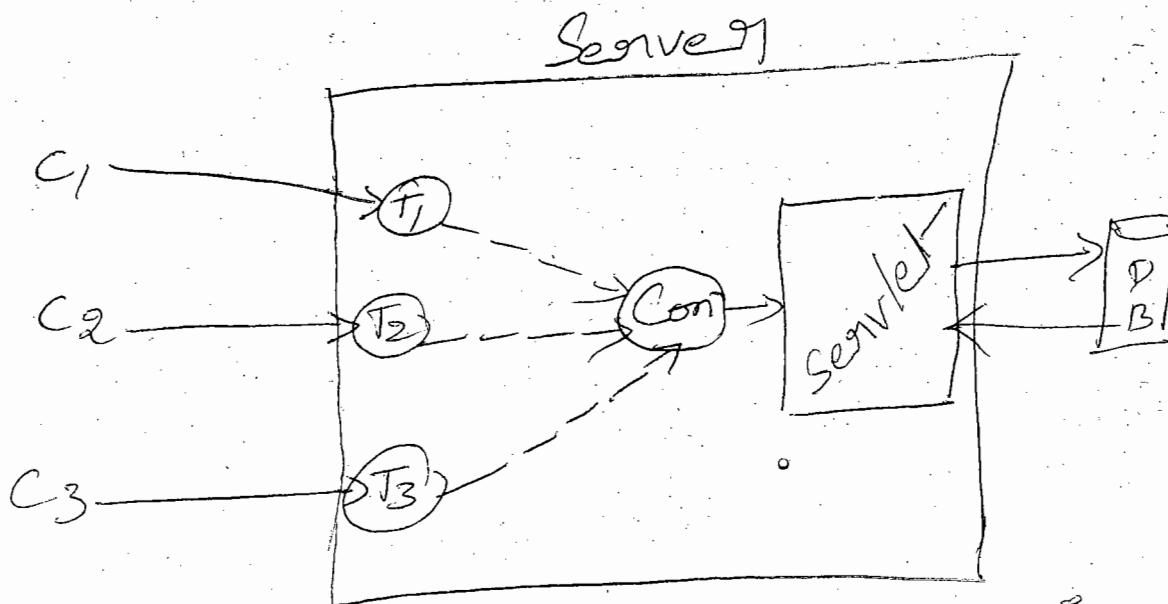
→ (i) open the database connection in the `init()` method, close the connection in `destory()` method & Remaining processing at `service()` method.

→ (ii) open the database connection, database operations & closing the connections & all operations in `service()` method.

→ (iii) Getting the connection from a connection pool to execute operations on database.

approach-(i)

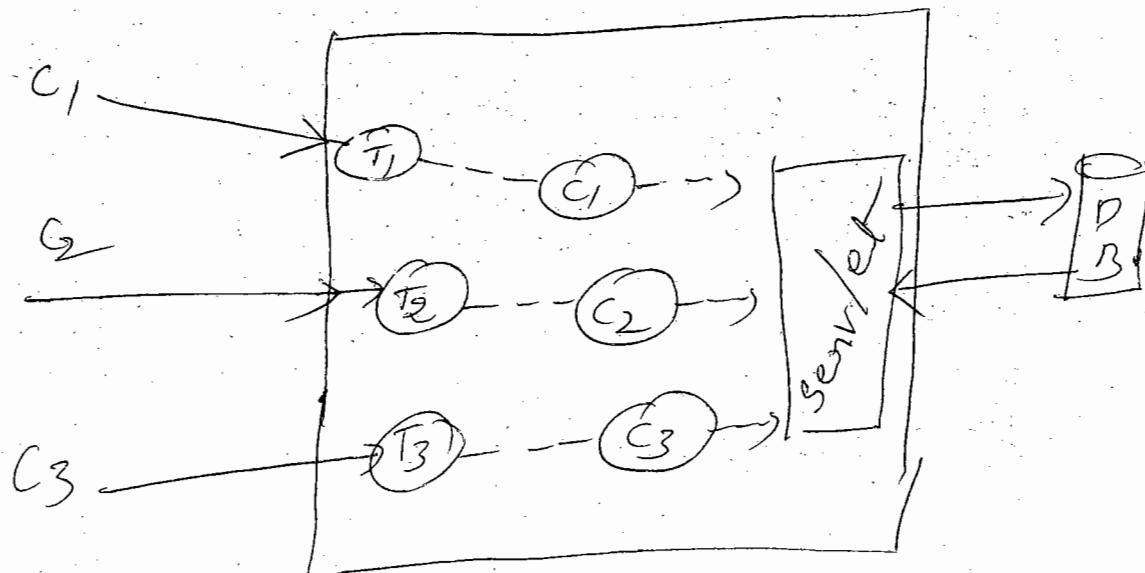
→ In this approach only one connection with database is opened & multiple threads servicing the client's requests will share single database connection.



The advantage of this approach is, only one connection is opened with database. So burden on the database server is reduced.

→ The drawback of this approach is only one connection is shared by multiple threads, so connection object becomes non-thread safe. It means if one modifies it other threads got ..

In this approach for each thread sent along the client request, a separate connection with database is created.



The advantage of this approach is each thread has its own connection, if one thread modify the connection object it doesn't effect on remaining threads, it means the connection object is thread-safe.

The drawback of this approach is burden on the database server will be increased.

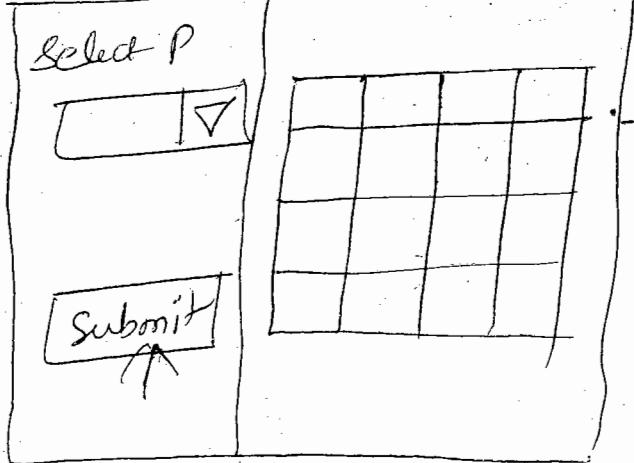
Note: To overcome the drawback of this approach II be got solved with 1.. technique.

Example

→ In this example, we divide a page into two frames on Browser & request is given ~~to~~ to the servlet from first frame & response will be displayed in the second frame.

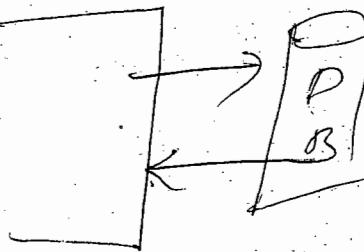
Main.htm

Select.htm



Servlet

~~Servlet~~



Logic: 1

- servlet Reads the parameter values & page number
- If client has not given, pageno parameter then servlet takes pageno as 0 & if client has given the page no, then servlet will take that pagenumber.

```
String sPageNo = req.getParameter("pageNo");
```

```
if(sPageNo == null)
```

```
{
```

```
    pageNumber = 0;
```

```
}
```

```
else
```

```
{
```

```
    pageNumber = Integer.parseInt(sPageNo);
```

```
}
```

```
=====
```

Logic 2:-

- we need to calculate the starting index by depending on page number.
- move the cursor directly to the starting index and then display the three records of that page.

- If we want to move the cursor directly to a particular Record than our ResultSet must be a

e.g. `ResultSet`

→ In JDBC, if we want to create non scrollable Resultset than we need to use type & mode parameters.

StartIndex = pageNumber * no of Records per page -
no of records per page + 1

rs.absolute(StartIndex);

i=0;

do

{

i++;

} while(i <= s) & rs.next();

logic 3 :-

→ Find total no of records available in the Table.

→ To get total number of records, we need count of the records from the Table.

→ Find total no of records by executing a select operation & store the count into the variable

ResultSet rs2 = stmt.executeQuery("Select count(*) from Emp");

rs2.next
→ (14)

"rs2.next()",

calculate the no of Pages required for displaying the links.

→ The number of pages are calculated by using
The Total no of records / No of Records per page.

→ If total number of records are greater than
number of pages * No of Records per page then
we need to increment number of pages by 1.

no of pages = Total No of Records / No of Records per page

if (Total/No of Records > (no of pages * no of Records per page))

{

 no of pages++;

}

Logic - 5 :-

---> Display the hyperlinks using anchor Tag &
for each hyperlink send a request to the servlet
with pageno = href no.

for (int i=1; i<=no of Pages; i++)

{

 pw.println (" + i + "");

}

code :- Page no 120

appl. q of the handbk.

Date
27/8/11 :- Servlet
 ↓
 with
 ↓
Connection Pooling

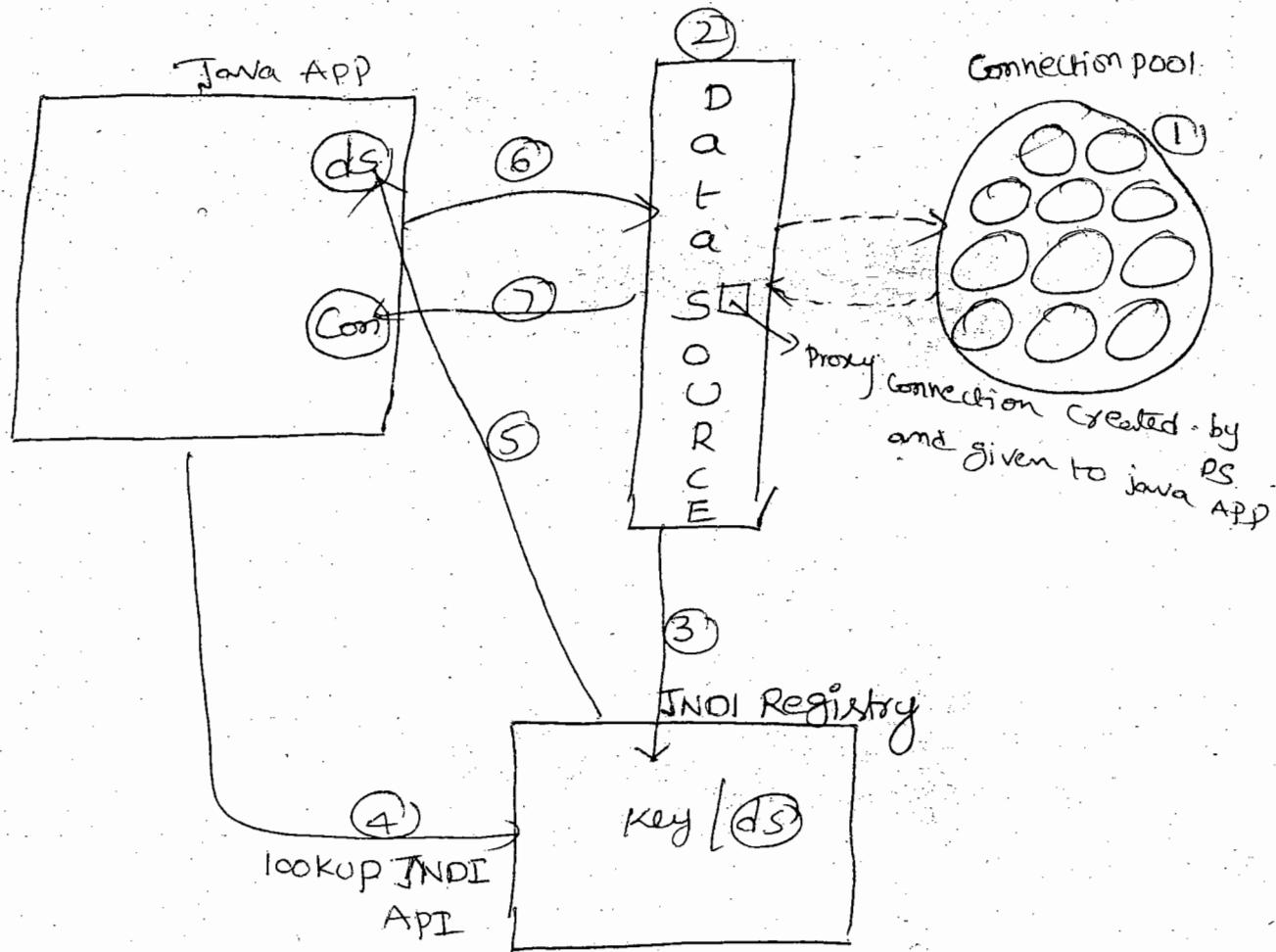
If we use DriverManager class to obtain a database connection in a java application like a desktop application or a servlet or an EJB etc, then we have the following problems

- (i) Java application is talking with database and exchanging the information like usernames, password, before opening the connection. It takes some amount of time.
- (ii) DriverManager provides a physical connection to the java application & it is a non-reusable connection. So each application has to open & close the connections separately. In this case some time is taken for allocating & deallocated the memories.

→ In order to overcome the above problems by not using DriverManager we have another native mechanism for java to database communication called as DataSource.

→ DataSource has a Connection Pooling implementation. So Java applications are going to

Connections with the database and these are reusable connections, so the application has no need to wait until the connection is opened & on the database server the burden is reduced.



- ① Server administrator creates a connection pool with a database.
- ② the administrator also creates a mediator to access the connection pool, we call this mediator as a Data source object.
- ③ The datasource object will be binded into the JNDI Registry by the administrator, with some key, we call this key also as a JNDI key.

(4) Java application uses JNDI API and it will search for the key in the Registry, we call this operation as a look-up operation

(5) Java application gets datasource object from the Registry.

(6) Java application, communicates with Data source to obtain a connection from the pool.

(7) Datasource internally gets a connection from the pool & creates a proxy connection & finally that proxy connection is given back to the Java application.

Note :-

datasource will return the original connection back to the pool, whenever the proxy connection is closed in the java application.

IMP points:-

(i) Connection pool contains dynamic behaviour. It means the pool has any idle connections for a long period of time, then automatically those are removed.

(ii) one connection pool is purely dedicated for one database. It means, it is not possible to store multiple-database connections into a single pool.

call

Proxy Connection is possible. It means, proxy connections can be created one after another but a physical connection, but not all at a time possible.

v) Each Connection pool has some capacity & limit and it never exceeds its maximum capacity.

Date _____

on 29/8/11

Finally Configuring a connection pool in weblogic 9.0

o the → Start the weblogic server

Start → programs → BEA products → weblogic

Server9.0

→ open the browser & type the following URL at the address bar.

http://localhost:7001/console

• Enter Username & password. (weblogic & weblogic)
click on Login button.

→ At left side, expand Services → Expand Jdbc → select

Data Sources → click on lock & edit button → New →

Enter the following details

Name: → Any Name (pool name)

JNDI Name: oracleJNDI

DatabaseType: oracle

→ next (click) → e

Enter the following Connection properties.

DatabaseName: ↑ Global database Name

HostName:

Port No:

UserName:

PSW:

Confirm Psw:

• click on next button.

→ click on Test Configuration Button → next →

Select Examples Server → Finish → click on
pool name (satyapool) → click on Connection Pool

→ Enter capacity values → Save → Activate
changes.

— — — — —
If a Servlet want to get connection from the
Connection pool, then in Servlet following JNDI
coding is required.

① Properties p=new Properties();

p.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");

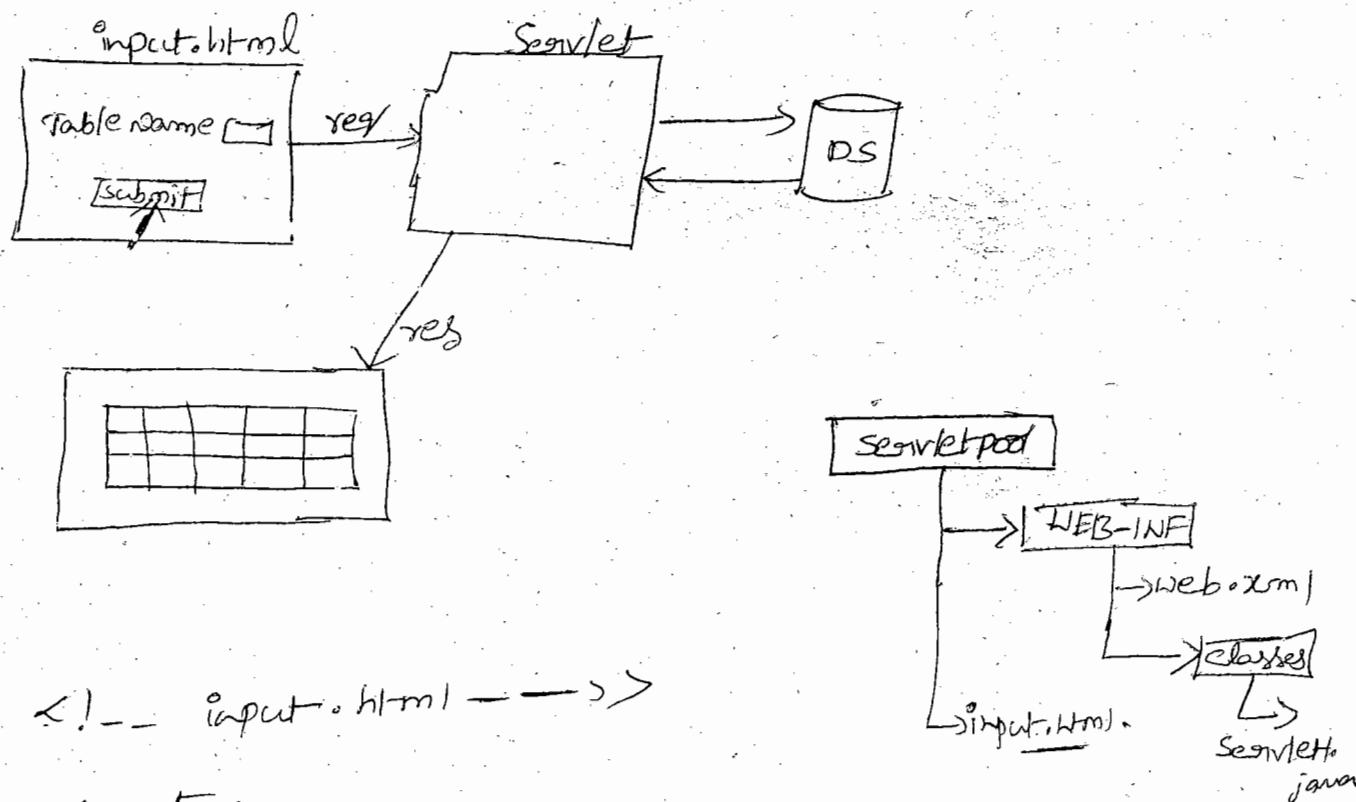
"");

p.put(Context.PROVIDER_URL, "http://localhost:
7001");

`DataSource ds = new (DataSource)();`

`Connection Con = ds.getConnection(); // ①`

Ex:-



`<!-- input.html -->`

`<center>`

`<form action = "Servlet">`

`TableName<input type = "text" name = "t1">
`

`<input type = "Submit" value = "click">`

`</form>`

`</center>`

`⇒ Servlet1.java`

```
import java.io.*;  
  "  " util.*;
```

```
import javax.naming.* // JNDI purpose
```

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

Extends HttpServlet

Public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException

{
try
{

String s1 = request.getParameter("E1");

// print P ---
=====

Statement stmt = Con.createStatement();

ResultSet rs = stmt.executeQuery

("select * from " + s1)

Gap com
using

PrintWriter pw = response.getWriter();

If (rs != null)

{

pw.print "

while (rs.next())

{

pw.print "|";
| |

pw.print " " + rs.getString(1) + " |";

"</td>");

pw.print " " + rs.getString(2) + "</td>"); |

```
    }  
    pw.println("vegetables");  
}  
else  
{  
    pw.println("no records in the Table / no table in  
    Database");  
}  
}  
}  
}  
else  
{  
    pw.close();  
    stmt.close();  
    con.close();  
}  
}  
catch(Exception e)  
{  
    System.out.println(e);  
    e.printStackTrace();  
}  
}
```

= = =
<web-app>
<Servlet>
<Servlet-name> pool </Servlet-name>
<Servlet-class> Servlet1 <-->
</Servlet>
<Servlet-mapping>
<Servlet-name>
<url-pattern>
</Servlet-mapping>

<S->
</url-pattern>

JAR file

C:\bea\weblogic9.0\Server\lib\Weblogic.jar

Note:

- * To compile the above servlet, we need to set some server given jar file in the class-path.
- * It is possible to compile a servlet by using jar-file given by one-server & we can run that servlet on another server.
- * In case of weblogic-server, the jar file given is weblogic.jar & it is available at the above specified location.

classpath = C:\bea\weblogic90\Server\lib\

weblogic.jar ; %classpath%

- weblogic.jar file is used for compiling JSP, Servlet & EJB applications also.
- Servlet-api.jar suitable only for Servlet applications.

Date 30/8/11 → Deploy the web-application root folder into C:\bea\weblogic90\Samples\domains\wl-server\autodeploy folder.

- Start the web-logic server.

.java

downloading

http://localhost:7001/servletPool/input.html

Applet to servlet communication:

www www www

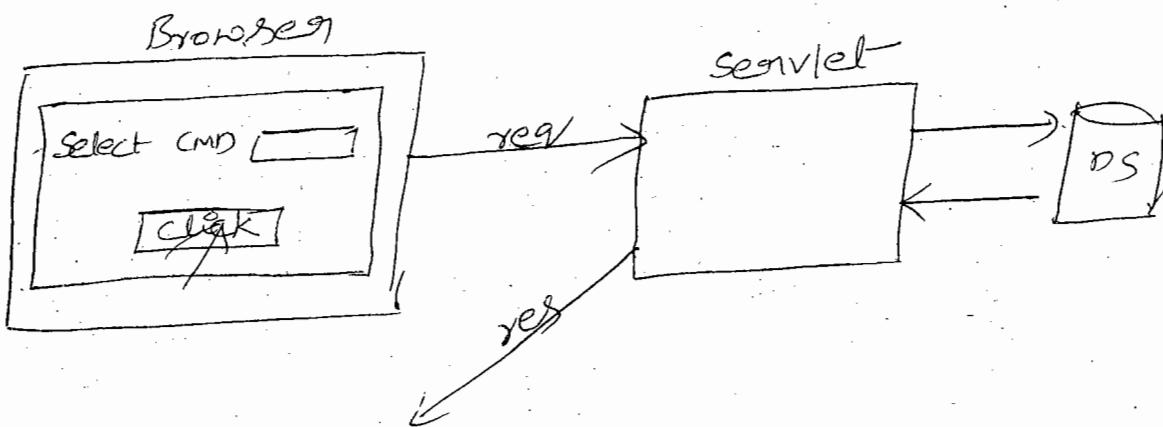
→ Generally we communicate from HTML to a servlet.

→ In case of HTML we are replacing with an applet.

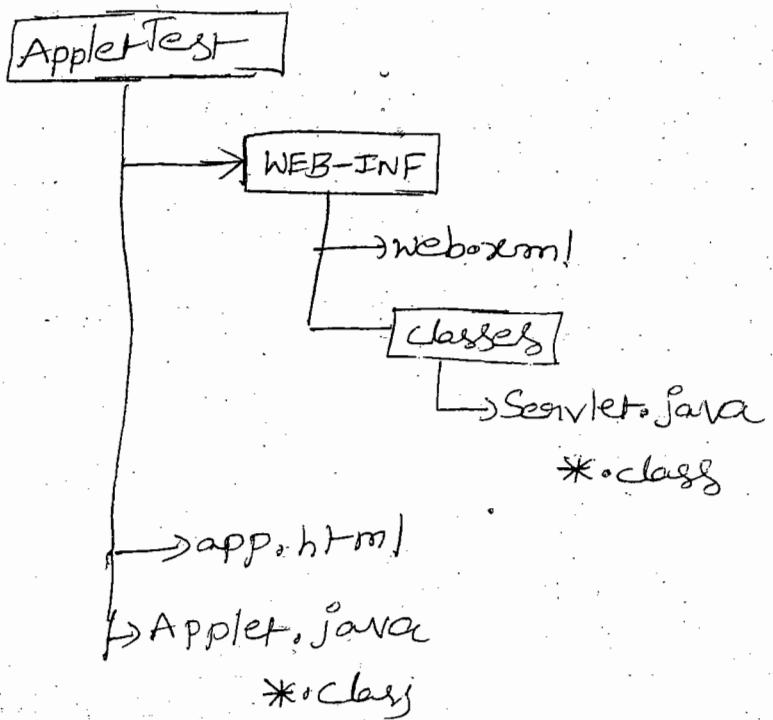
→ In applet to servlet communication, applet provides presentation on to the browser by accepting input values from the client.

→ In order to integrate or communicate from an applet to servlet we need to use a class called URL.

→ URL is a class given in `java.net` package.



Directory Structure



1) <!-- app.html -->

Applet code = "TestApplet.class" width=500
height=200>
</applet>

2) Applet.java

```

import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;

public class Applet extends Applet implements
  ActionListener
  
```

Curly braces are shown above the code, indicating the class definition. Brackets are placed around the import statements and the extends clause. Braces are also placed around the implements clause. A brace is placed after the implements clause, spanning the entire class definition.

Annotations:

- A brace is placed around the first three import statements, with the text "interface mechanism" written next to it.
- A brace is placed around the last two import statements, with the text "doesn't apply to Packages" written next to it.

Label l

Button b

TextField tf

public void init() {
 }
 l = new Label("Enter");
 t = new Textfield(20);
 b = new Button("Get Data");
 add(l);
 add(t);
 add(b);
 b.addActionListener(this);

- ⇒ two types of events
- Low level → Remaining only
 - Syntactic
-

public void paint(Graphics g)

setBackground(Color.Green);

public void actionPerformed(ActionEvent ae)

try {
 String s = tf.getText();
 URL u = new URL("http://localhost:2011/");

Applet follow by default

Flow Layout

Frame follow by default

Bordered Layout

}
 catch(Exception e) {
 AppletTest/Svr? cmd=+S)

e.printStackTrace();

AppletContext ct = getAppletContext();

ct.showDocument("u, "self");

Servlet.java

```
import java.servlet.*;
import java.io.*;
import java.sql.*;

public class Servlet Extends GTS
{
    public void service (ServletRequest req,
                         ServletResponse res)
        throws ServletException,
               IOException;
    {
        try
        {
            String s1 = req.getParameter ("cmd");
            PrintWriter pw = res.getWriter();
            Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con = DriverManager.getConnection
                ("jdbc:odbc:oradsh",
                 "scott", "tiger");
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery (s1);
            pw.println ("

||
||
||


```

```
                pw.println ("
```

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

```
Connection con = DriverManager.getConnection
```

```
("jdbc:odbc:oradsh")
```

```
"scott", "tiger")
```

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

```
ResultSet rs = st.executeQuery (s1);
```

```
pw.println ("

||
||
||


```

```
    pw.println ("
```

pw.println("jeremy just visited").
((n+1)+(1+(2+3)))
((1)+(3))-

pw.println("try")

3

pw.println("Tables")

pw.close();

rs.close();

con.close();

3

catch(Exception e)

e.printStackTrace()

2

| http://localhost:2011/AppletTest/appointm)

394

dsh

Date
31/8/11

→ ↗

in

are

RequestDispatcher interface (included Servlet 2.2)

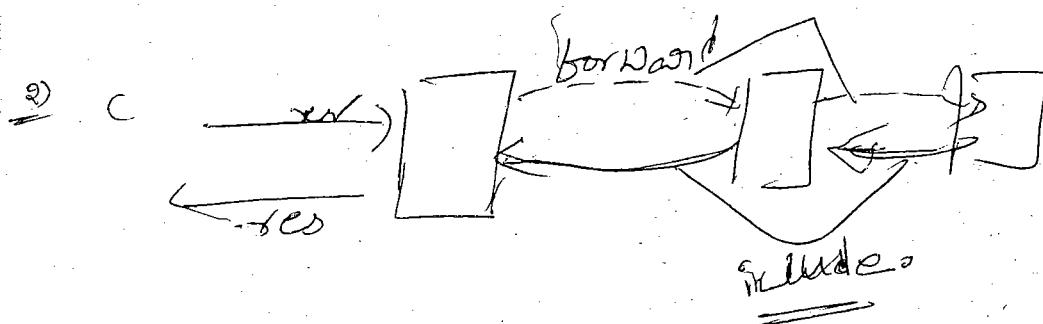
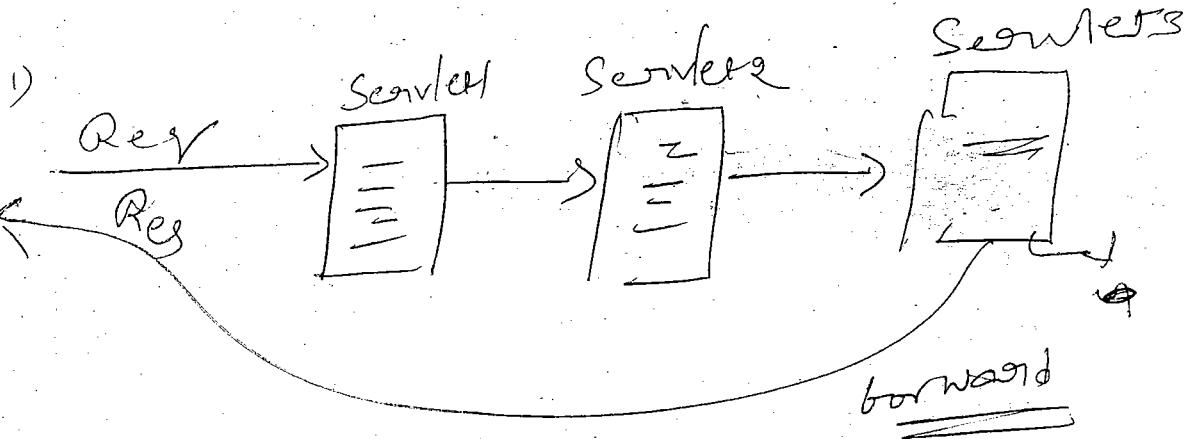
- In Servlet 2.0 & 2.1 versions, we have a concept called servlet chaining. It means moving the control from one servlet to another servlet.
- Instead of defining the entire logic for a single servlet, we can divide the logic into multiple servlets, and we can execute them one after another. We call this concept as chaining or linking.
- In Servlet (e.g. Servlet 2.1), this linking b/w servlet is done by administrator at server side. But in Servlet 2.2, the servlet chaining concept is moved from ~~admin~~ administrator side to program side by introducing RequestDispatcher.
- By using RequestDispatcher, we can perform only two operations:
 - 1. We can forward request from one servlet to another servlet.
 - 2. We can include the response. If we want to include the response of the

in Request Dispatcher Interface 2

are provided -

they - (1) forward (req, res)

(2) include (req, res);



forwarding

→ While forwarding a given request from one servlet to another servlet, the last servlet in the chain will produce response back to the client.

→ While forwarding a client request from one servlet to another servlet, there will be no information given back to the client. It means the forwarding technique works at serverside and does not intimate the client browser.

→ While forwarding, a client request is forwarded from one servlet to another. It means one pair of

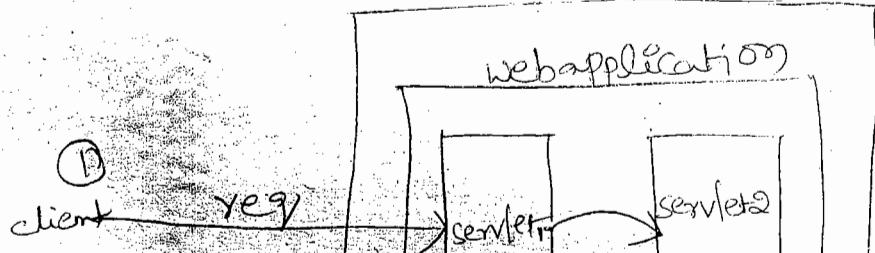
... contains

↳ Same request & response objects are continued
in the entire forwarding chain. (2)

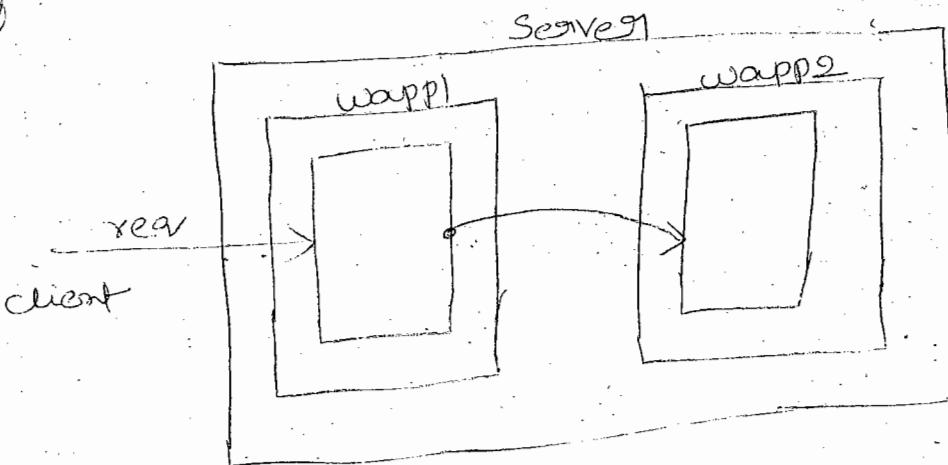
→ while forwarding a request from one servlet to another servlet, the request parameters are also automatically forwarded to the destination.

- → Forwarding a request from one servlet to another servlet permanently. It is nothing but moving the ctrl permanently to one servlet to another, so it is not possible to forward to multiple servlet from a single source servlet. (3)

- → while forwarding, from a single source servlet the destination resource must be Java enabled resource only (servlet or JSP, or HTML). =>
- → while forwarding a request, both source & destination servlets must exist either within the same webapplication or different webapplication at the same server - but it is not possible to forward a request from one server to another server.



one Ser
meting a
situation



it to
g but

vlet to

and to
vlet

Se
enabled

=> including
in another

stage
within
application
etc
whether the client by the source servlet.

→ while including the response of one servlet into another servlet, the response will be partially generated by source or partially generated by destination. The total response will be given back to the client by the source servlet.

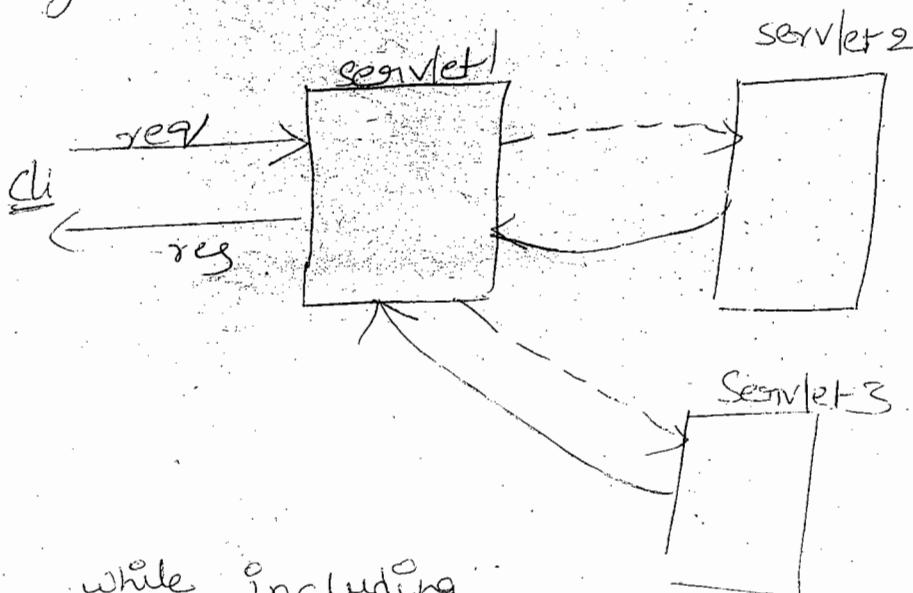
→ while including the response, the include operation is done at serverside and no intimation is given back to the client so the client is unaware about the include operation.

→ while including, the webcontainer will only generate request & response objects and

→ while including the response the ctrl is temporarily shifted from a source to a destination and again the ctrl goes back to the source with the generated response. so it is possible to include multiple destination response into a single source.

(1)

client



(2)

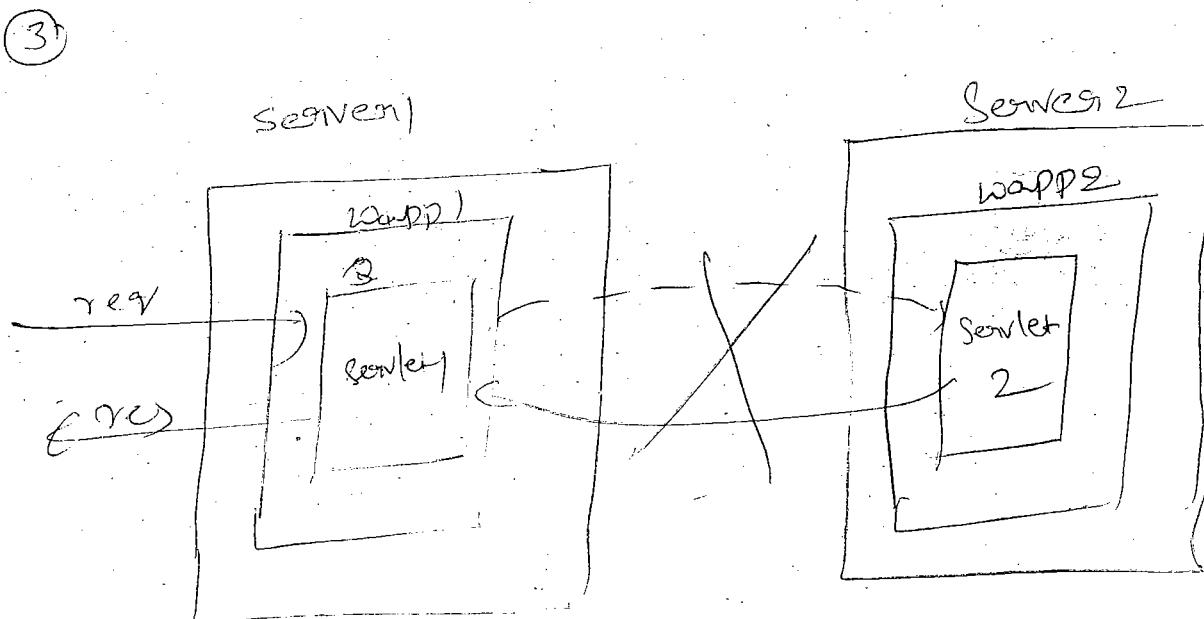
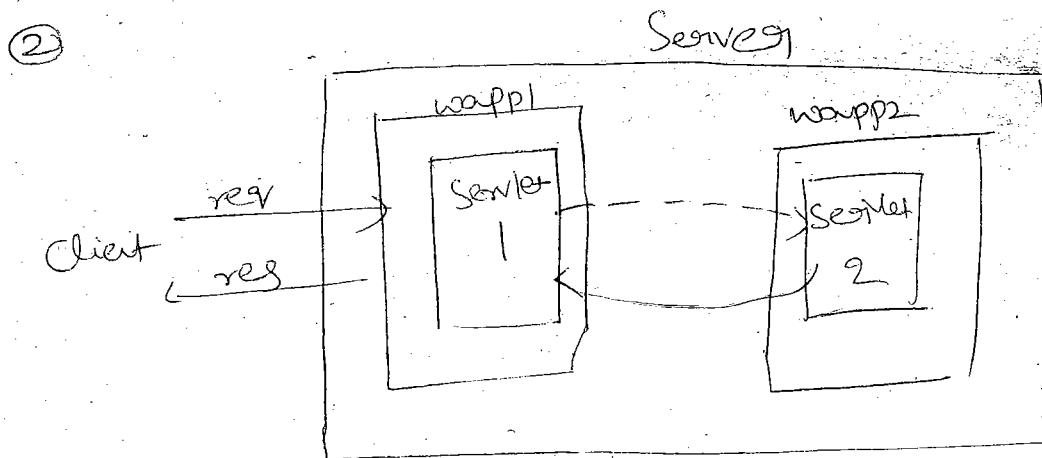
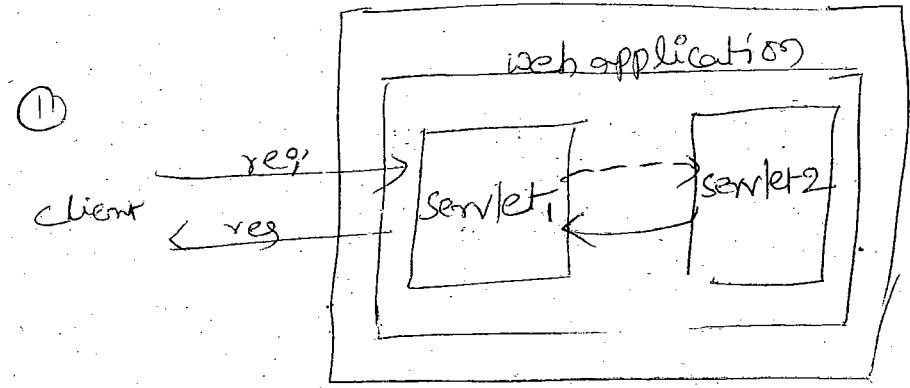
client

→ while including the response, the ctrl is temporarily shifted to the destination along with ctrl the request parameters are also transferred to the destination automatically.

(3)

→ while including, destination resource must be a Java enabled resource only.

→ while including the response, both source & destination resources must be available either in same webapplication or in different web application of the same server but it is not possible to communicate across different servers.



Obtaining RequestDispatcher Object

Date
31/9/11

- ① RequestDispatcher rd = ctx.getNamedDispatcher("String Servlet")
method → return
② RequestDispatcher rd = ctx.getRequestDispatcher("String path")
→ rebu
③ RequestDispatcher rd = request.getRequestDispatcher("String path")
→ a begin

approach - ①

Ex:-

C

→ If we want to forward or include, we need a request dispatcher object in the source servlet & it should point to destination servlet.

→ In this approach, we can get request dispatcher object through the logicalName of the destination servlet.

→ among the all approaches, this approach is simple way of getting a RequestDispatcher Object because we no need to pass any path as a parameter

approac

→ In

meth

→ s

path

objec

Ex:- Servlet

service()

Dummy
newbie

→

3/9/11 approach-2

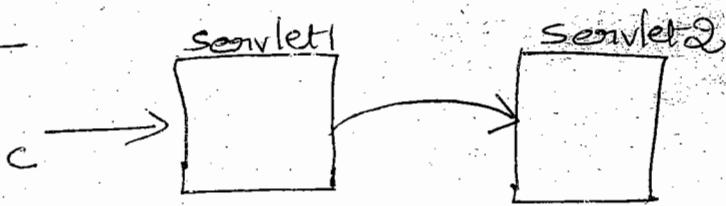
In this approach we use getRequestDispatcher method given by context object.

Servlet
runner
catcher

→ In this approach we need to pass context relative path, to get RequestDispatcher object.

→ Context relative path means the path must begin with "/" and root directory.

Ex:-



web.xml

servlet1 → /srv1.

servlet2 → /srv2.

servlet1

service()

RD rd = ctx.getRequestDispatcher("/wapp1/srv2")

rd.forward(req, res)

approach-3

→ In this approach, we call getRequestDispatcher() method given by ServletRequest object.

→ In this approach, we need to pass request relative path as a parameter, to get RequestDispatcher object.

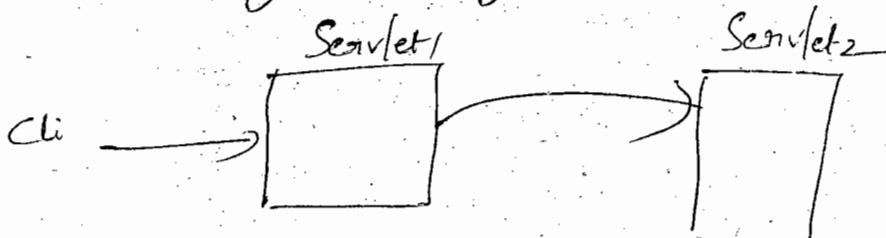
→ Request relative path means we need to

my
note
book

the URL context and we

→ while writing the Request relative path, if we want to come back any directory then use "..."

→ It may or may not begin with '/'.



(3)

web.xml

Servlet1 → /srv1

Servlet2 → /srv2

Servlet

service()

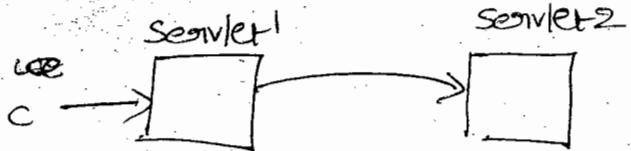
RD rd = req.getRequestDispatcher("srv2")

rd.forward(req, res);

handl

Page n

⇒ The following example is to differentiate the context relative & request relative path



web.xml

Servlet1 → /aa/bb/cc /srv1

Servlet2 → /aa/bb/mm /srv2

Servlet1

service()

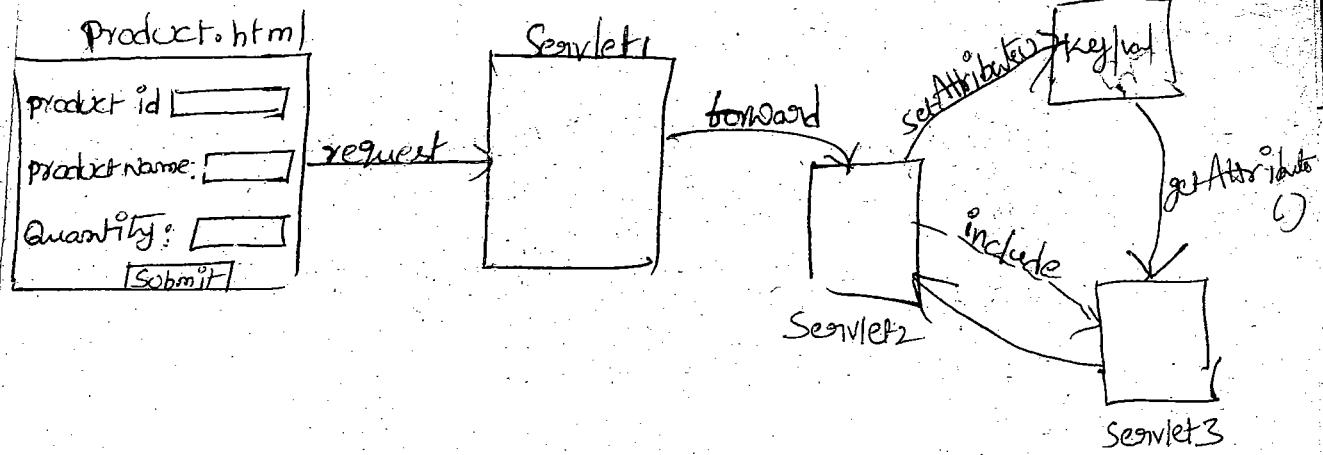
RD rd = ctx.getRequestDispatcher("/wapp2")

rd.forward(req, res);

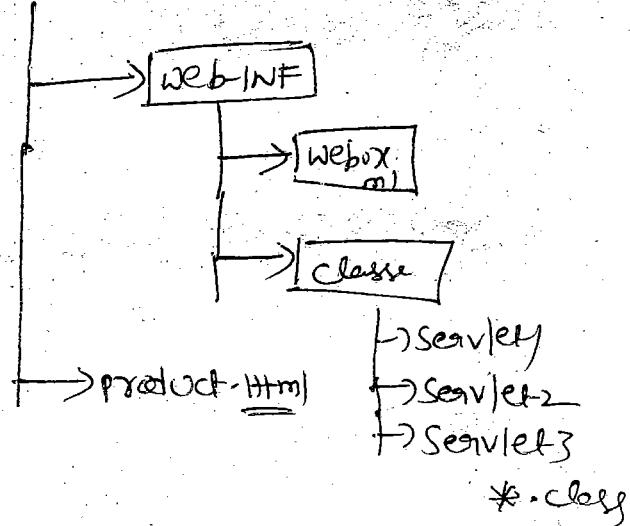
(or)

aa/bb/mm
srv2

... Dispatcher / wapp2



RequestDispatcher APP



handbook

Page No: 127, ex: no: 13

"Srv," → correct

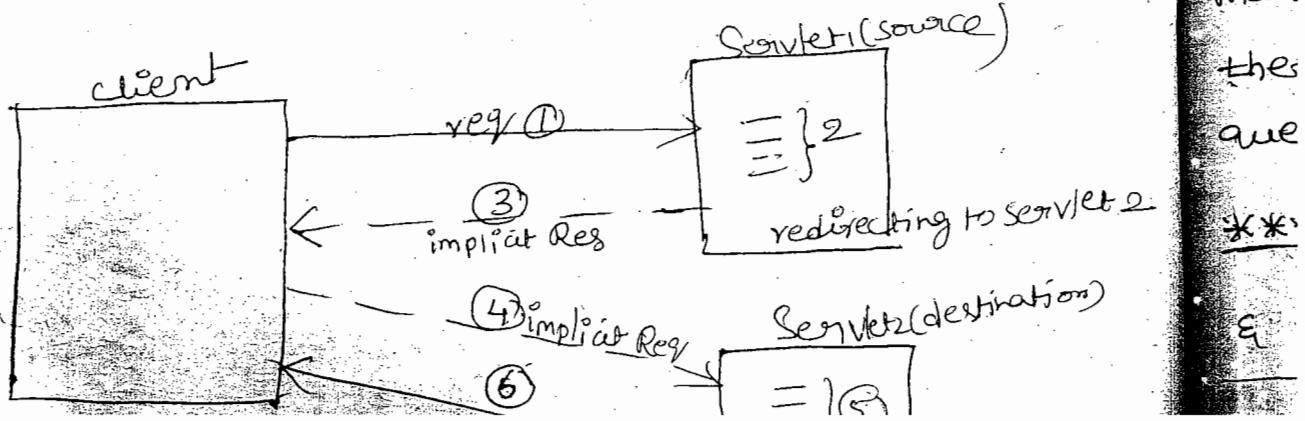
"/Srv," → not correct

"./Srv," → correct, specifying wrong webapplication,

date
4/9/11

Servlet Redirection

- Servlet Redirection is also belongs to chaining of servlets.
- If a client request is redirected from one servlet to another servlet i.e., from a source to destination than first of all the control implicitly goes to the browser, intimates the client and then the control is shifted implicitly from browser to second servlet (or) destination.
- At the time of redirecting a request the redirection is not done only at server side. First of all the client is intimated and then after, the control goes to the destination.
- In a redirection technique, the redirection information is provided to the client also.
- In order to do redirection, we need to call a method sendRedirect(), given by HttpServletResponse interface.



URL in address bar will be changed from old one to new one.

For example, if we type `www.bea.com` in the address bar and when we send a request then the request is ~~`www.bea.com`~~ redirected to `bea` website to oracle website and the address bar is changed to `www.oracle.com/bea/index.htm`.

www. grade . com / bay / index . htm

control
is the Generally we use Redirection when an old website
+ simple URL is changed to a new URL or when we are
destination communicating from one server to another server.

section → In case of Redirection minimum 2 requests are generated from the client side. explicitly

- Call a Response

 - (i) Given by the client explicitly
 - (ii) Generated implicitly while redirection.
 - (iii) in Redirection, only ctrl will be redirected from a source to a destination, but not the data automatically. So if we want to redirect the data also then, we should add explicitly in the form of a query String.

* * * the following are the differences b/w forwarding & Redirection techniques

Forwarding	Redirection
o mail, innovation, the	

and the information given to the client.

→ During Redirection operation, the information is given to the client and then redirected to the destination.

username
password
[su]

→ In forwarding, one request only forwarded to the destination. So container creates a single pair of request & response objects.

→ In Redirection, implicit a new Request is generated so a minimum of two pairs of request & response objects are generated by the container.

→ In forwarding both the control and the data are forwarded to the destination (by default).

→ In redirection, only control is redirected, but not the data (by default).

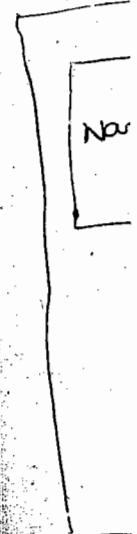
→ In forwarding, the destination resource must be java enabled resource only.

→ In Redirection, the destination resource can be either Java or Non-Java resource also.

Ex:-

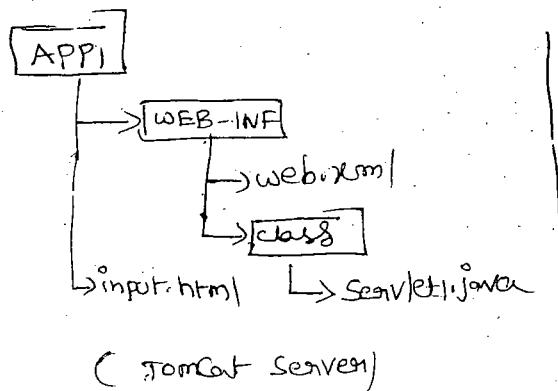
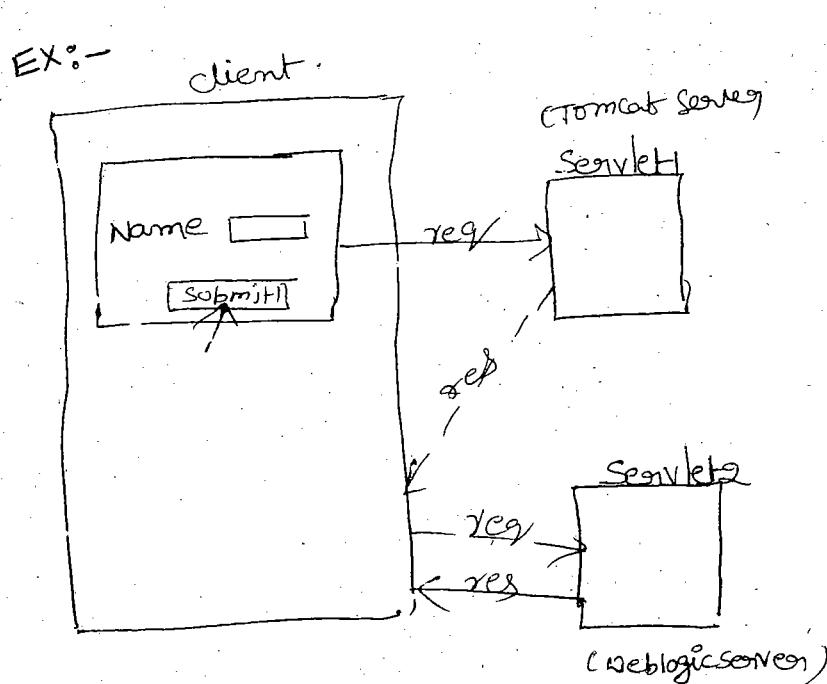
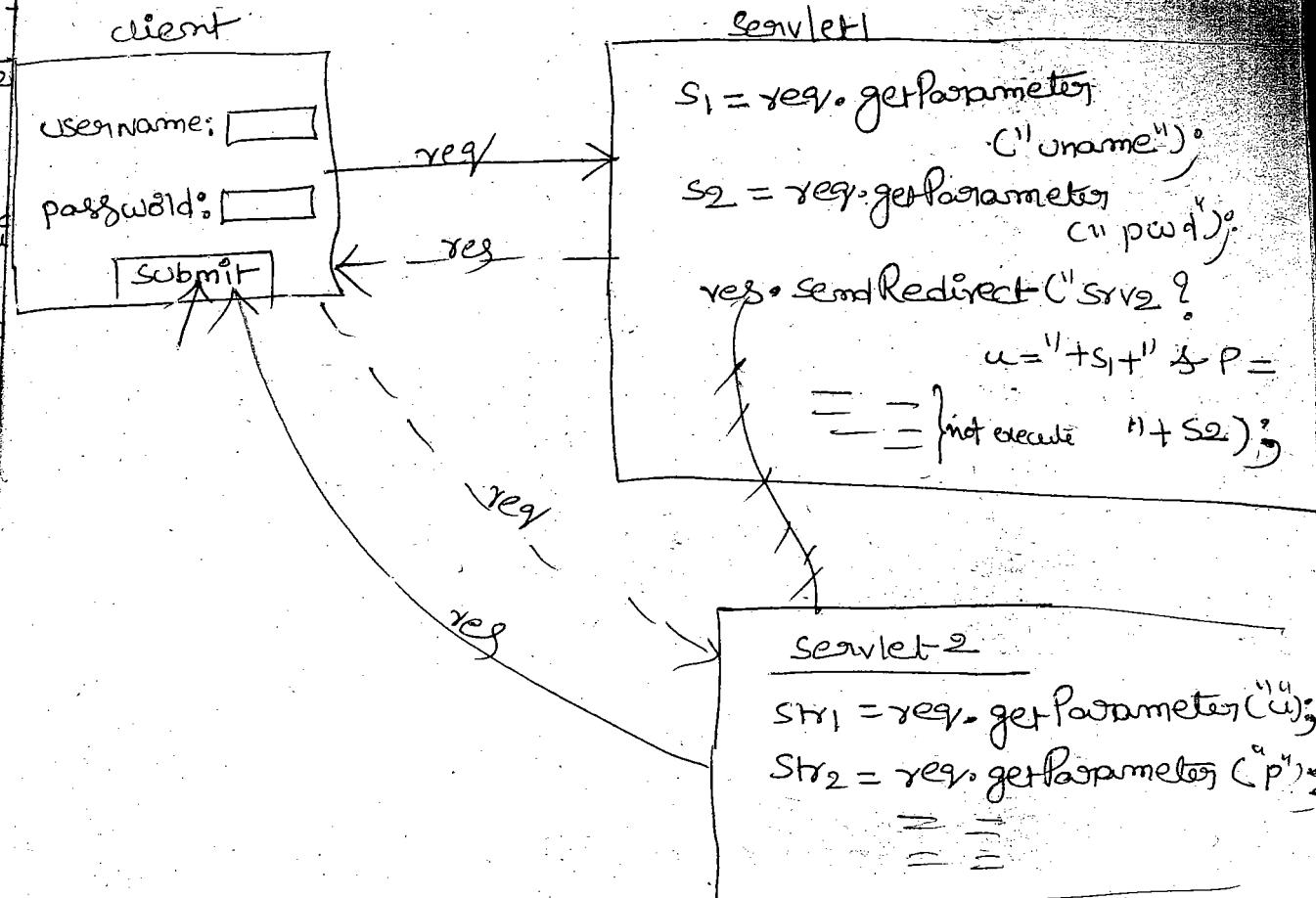
→ In forwarding, both source & destination resources must run within the same server. It is not possible to communicate across the servers.

→ In redirection, it is possible to communicate either within the server or even across servers also.



Note:-

while re-directing the requests, if we want to attach the request values also, then we should explicitly add ~~above~~ ~~the~~ ~~the~~ above values in the URL or destination like the following



Tomcat Server / App

weblogic

```

<!-- input.html -->
<center>
<form action = "srv1">
Enter Name: <input type = "text" name = "uname"><br>
<input type = "Submit" value = "click">
</center>

```

Servlet1.java

```

import java.io.*;
import javax.servlet.*; HTTP
import javax.servlet.HttpServlet.*; Servlet
public class Servlet1 extends HttpServlet
{
    public void doGet(HttpServletRequest req,
                      HttpServletResponse resp)
        throws ServletException, IOException
    {
        String s1 = req.getParameter("uname");
        resp.sendRedirect("http://localhost:7001/APB2/srv2?p1=" + s1);
    }
}

```

import
import
import
pw

Step 3

→ C

→

→

→

→ .

goes

to

so

```
import javax.servlet.*;  
import javax.servlet.http.*;  
import java.io.*;  
  
{  
    public void doGet(HttpServletRequest req, HttpServletResponse  
        throws ServletException, IOException {  
    }  
}
```

String s1 = req.getParameter("P1");

PrintWriter pw = res.getWriter();

pw.println(" from weblogic "+v1);

pw.close();

}

Step 8

- Deploy app, into tomcat & App2 into weblogic server.
- start both tomcat & weblogic servers.
- open the browser & Type the following request

http://localhost:2011/APP1/input.htm

→ when we click the Submit button then the request goes to Servlet1 and it is redirected from Servlet1 to Servlet2 that is presenting in weblogic server.
So the Address bar changed in the following way

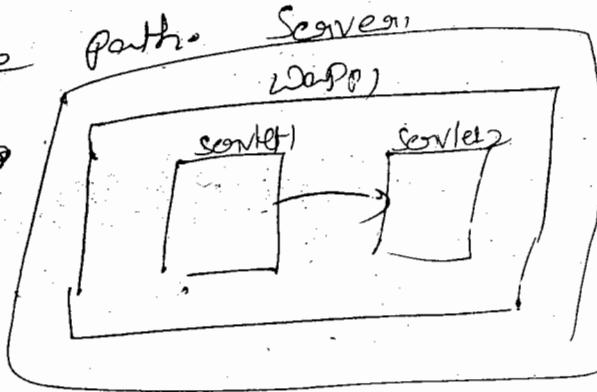
http://localhost:7001/App2/Serv2?P1=Satya.

IMP points

- (i) Redirection works only at HttpServlet. (c)
 Cause sendRedirect() is given by HttpServletReq Serve
 but not ServletResponse.
- (ii) Forwarding works for both GenericServlet &
 also HttpServlet. because to forward() method
 we can pass servletRequest, servletResponse or
 HttpServletRequest, HttpServletResponse.

⇒ while Redirecting, we need to provide the
 Path like the following.

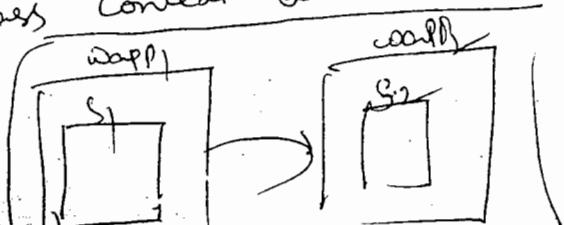
- (a) if source and destination are available within same webapplication then we need to pass request relative path.



ref. sendRedirect("srv2")

- (b) if source & destination are available within the same server but diff webapplication then

We need to Pass context Relative Path



Date

→ a

→ whe

→ stae

→ so

→ to

→ th

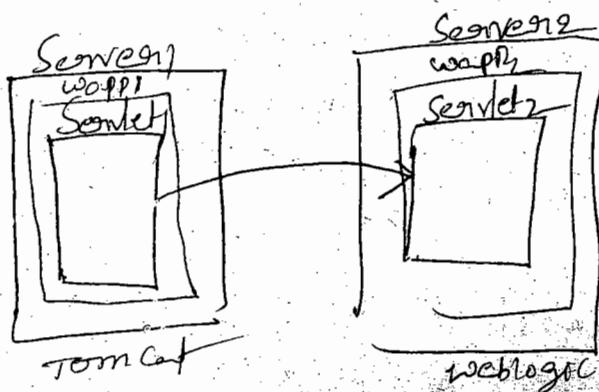
→ ion

→ e

8

Servlet (c) If servlet1, & servlet2 available at different servers than we need to pass absolute path.

Servlet &
method
use (or)



res.sendRedirect("http://localhost:7001/
wapp2/srv2")

e the

Date: 6/9/11

Within (IMP)

pass

Session - Tracking

- Session is nothing but it is a time-interval b/w a login & logout.
- For example after entering username & pswo when we click on login button, then one session starts & it will be continued until we say logout.
- by default HTTP protocol is a stateless protocol so, managing the session is not possible.
- In order to manage the sessions, we need to convert the HTTP behaviour from stateless into statefull.
- In order to convert HTTP protocol from stateless into statefull we need session-tracking.
- Session-Tracking is the combination of both session management & state management.

recognize that these unique memory are given to same-client only.

A to
quest &
session-

- State management means, at server side information related to a client should to be stored so that the data is sharable across multiple requests given by the client.

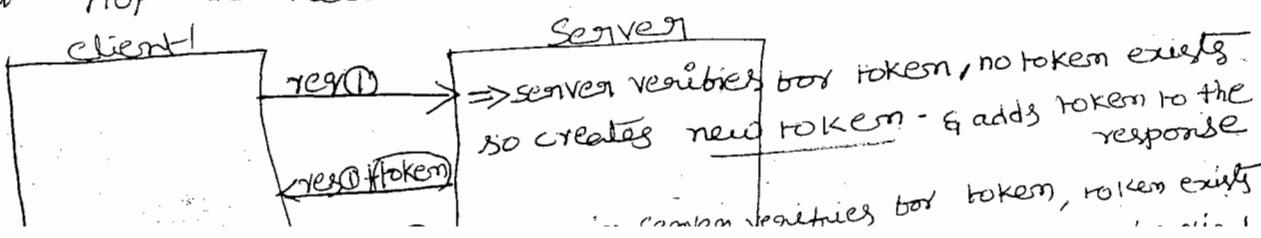
Session Management:-

In session management, server first verifies, whether the client is a new client or old client by observing a token in the request.

- If no token exists than server will treat that the client is new and creates a token for that client.

→ while sending the response back to the client, than along with response that token is also sent back to the client.

- when a client is next-time sending the request to a server than along with the request, the token is also sent. To the server so server recognizes that the client is existing but not a new client.



request & response. so by using this token exchanging technique, session-management is done by the server.

server-side

be stored state management:-

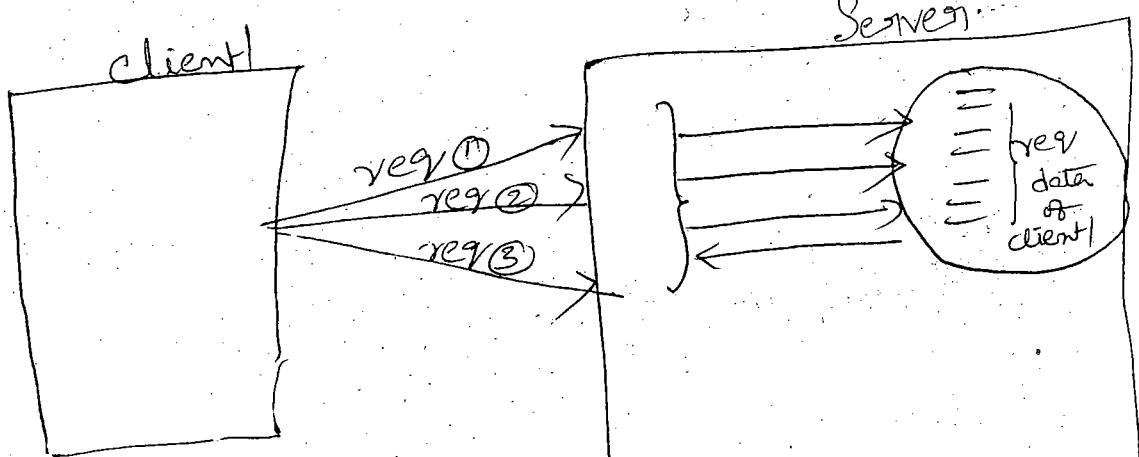
Multiple requests :-> when first-time a client request is given to a server then along with new token, an object is also created, for storing the information related to a client.

Subsequent requests :-> when the same client is sending next request clients to the server, then the information of the client will be accessed from that object.

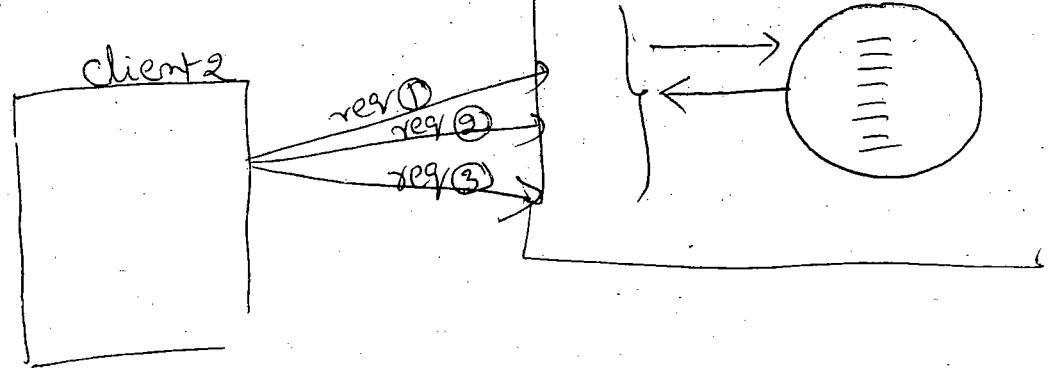
in the will be accessed from the client-side
For each client, one object is created at server-side
by maintaining the data of the client.

will
a token
The number of clients requested a server is equal to the number of objects created at server. these objects are for state management.

the
server
is



he
reque
t is so
ting



lets
the
use
exists
client

For complete session-tracking i.e., for

Session management & State management, at 5)

server side a token & an object both are created at serverside. Here token is for session mgmt and an object is created for state mgmt.

Date

8/9/11

Session - Tracking Technically

6)

Session - Tracking means, creating a session-ID & session-object for managing session & state of the client.

Technically, session-ID is transferred b/w server & client in the form of a cookie.

From Server to client the cookie will be transferred by storing in ~~request~~ response header and from client to Server, the cookie will be transferred by storing in request header.

→ The cookie created by Server stores session-ID into it with the key called sessionid.

→ A cookie is created at server-side & stored on the client browser.

Diagram:

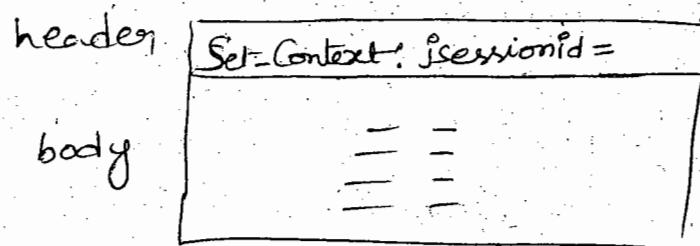
client $\xrightarrow{\text{req①}}$ server

1) session id created + session object-created

2) session id stored in session object

3) request- processed

5) cookie is added to response object, in header part



6) Finally, response is given back to client.

client req + cookie → server

⇒ server identifies the client
that client is existing client

⇒ uses session Id & session
object, already created by
the client.

Session - Tracking with Servlet API :-

In Servlet we have following two techniques for
Session - Tracking They are

- ① HttpSession (Interface)
 - ② cookies
 - ③ hidden field
 - ④ URL rewriting
- Partial Session tracking (SessionId + Session object)
Complete Session Tracing (SessionId + Session object)

1) HttpSession Interface

- HttpSession technique is for both session management & state management.
- Using Servlet API, we can associate the state with the sessions. It means we can store the client information in a Session object & we can read that information whenever it is required.
- As a Servlet programmer, we can perform state management operations & session management will be taken care by the Container.
- In order to get session object for storing state of a client, we need to ask the container to give a session object. This can be done by calling `getSession()` method given by HttpServletRequest Interface.
- While asking for a session, container will create session ID also. for managing session of the client.

→ In a servlet, we need to store the session into HttpSession Interface.

→ These are the different ways to store session object.

9/11/11 → In order to work with HttpSession
we need to extend our servlet class from HttpServlet.

- As a programmer, we can only store & read the data using a session object, but we can not create session ID & session object. The creation of session ID & session object will be taken care by the container.
→ In order to work with session object, first of all we need to get the session object by using any one of the following 3 approaches.

HttpSession session = request.getSession()

 " " = " " " (true);

 " " = " " " (false);

In the above three approaches, approach 1 & 2 are exactly same. The functionality is, when we call getSession() method, container first verifies whether these request is given by an existing client or a new client.

If the client is existing then already associated session of the client is returned. If the client is new then container creates a new session ID & new session object and then that session object is given back to the servlet application.

In approach 3, container verifies whether the client is existing client (or) new client. If existing then already associated session object is given to the client. If new then the container returns null value.

in approach 3, new session is not created by the container.

clies

cont

=

clies

mu

clies

Accessing session object:-

→ Accessing a session object is nothing but, storing the data into a session object (or) reading the data from a session object.

→ If we want to store the data in a session object (or) if we want to read the data from a session object then we need to use attributes.

In servlet API, we can store attributes (or) we can read attributes through the methods related to attributes API.

String Object
 ↓ ↓

- (i) setAttribute (Key, value);
- (ii) getAttribute (key);
- (iii) removeAttribute (key);
- (iv) getAttributeNames();

⇒ In servlet api we can store attributes into 3 objects.

(i) request (ii) session (iii) context

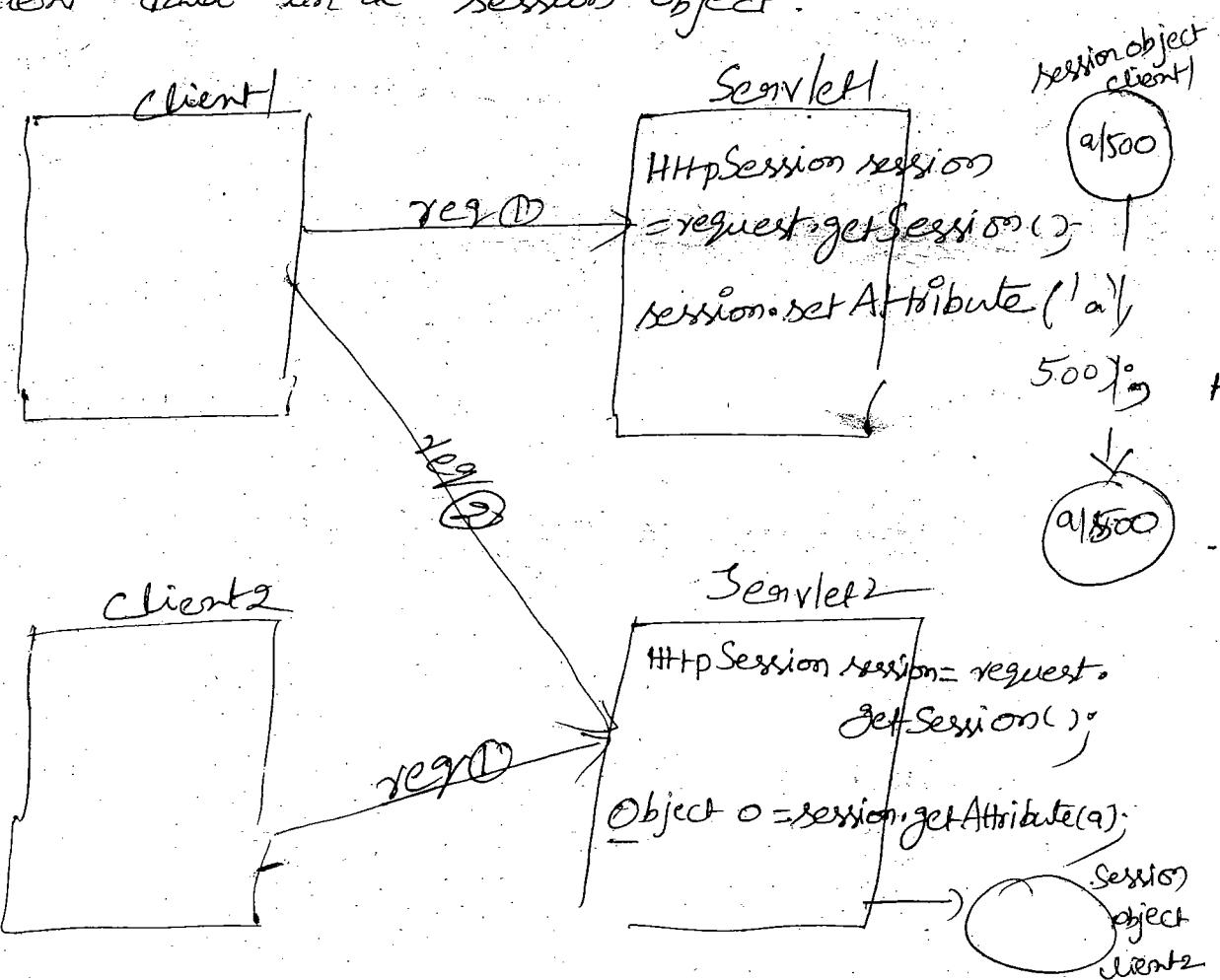
⇒ In request & context objects, we can store attributes and also parameters.

⇒ In a session object we can store attributes but not parameters.

⇒ If we store some data in the session

Container.

⇒ If we want to use (or) share the data of the client when the client is visiting the server for multiple ~~times~~ ^{times} then we need to store that client data in a session object.



→ According to the above diagram, when client 1 is sending the request to servlet 1 then the object o contains the value 500. and when client 2 is sending the request to servlet 2, the object o containing the value null. because for new client a new session object is created and in that session object there is no attribute called a. so the object o containing null.

→ whenever a session object is created by the container, the container will store some additional information into session object.

state(new or old), creation-time, ideal-time etc

→ This

→ In order to read the additional information stored in the session object by the container we need to call some session-management methods given in HttpSession interface. In HttpSession Interface we have some session-management methods and some state-management methods.

↑
e

Session management methods:-

(3)

- 1) getId() 2) isNew() 3) getCreationTime()
- 4) getLastAccessedTime() 5) getMaxInactiveInterval()
- 6) setMaxInactiveInterval() 7) invalidate()

This
are
one

System
on

State management methods:-

(4)

setAttribute(K, V);

getAttribute(K);

removeAttribute(K);

getAttributeNames();

date
0/9/10

①

getId();

→ This method is used to get the sessionID stored in a session object. The sessionID will be a large string value created by the container.

String str = session.getId();

pw.println(" sessionId = " + str);

2) isNew();

time
the
b/w

one min

newSession → returns true
existingSession → " false.

```
if(session.isNew())
{
    // logic - 1.
}
else
{
    // logic - 2.
}
```

(3) getCreationTime();

This method is going to return session creation time in the form of milliseconds (ms). These milliseconds are calculated from 1970 Jan 1st 00:00 hrs.

We need to convert the milliseconds value into System-date & time format by using Date class given in ~~java.lang~~ java.util package.

```
long ms = session.getCreationTime();
```

```
java.util.Date d = new java.util.Date(ms);  
pw.println(d.toString());
```

(4) getLastAccessedTime()

This method will returns the previous request time within a session of client. This method also returning milliseconds which are calculated from 1970 Jan 1st 00:00 hrs.

The first & second request the session creation time, last-accessed time both are one & same. From third request onwards we will get a difference b/w session creation time & last accessed time.

```
long ms = session.getLastAccessedTime();
```

Setting Session - TimeOut Period

→ while working with sessions at server-side, for each session there is some expiration-time set by the container.

→ By default most of the container will set expiration time as 4hr.

→ If a request gap from client to another is exceeding the expiration time-set than the session of the client at server-side will be removed.

→ we can increase or decrease the expiration time-set by the container. we can do it in any one of the following 2 ways.

(i) programmatic approach

(ii) declarative "

→ In programmatic approach, we need to call `setMaxInactiveInterval()` method inorder to set the expiration time. In this case the input value should be in seconds.

session.setMaxInactiveInterval(300);

→ In declarative approach, if we want to set the session Time-out period, then we need to configure `session-config` element in `web.xml` file.

web.xml

<web-app>

<session-config>

timeout>

or each
the
matice declarative approaches, then programmatic value is
considered by the container.

The following are the two differences b/w programmatic
& declarative approach

(a) In programmatic approach, the input value will be
in seconds. So we can set least session time-out
as '1' second, but in declarative approach, the least
session time-out period will be 1 minute.

(b) In programmatic approach we can change the expira-
tion time from one servlet-to-another servlet,
in declarative approach one time-period we can set
it is commonly applicable for all servlets

=> getMaxInactiveInterval() expired time

This method is used to get the session
in the form of seconds.

long s = session.getMaxInactiveInterval();

=> invalidate();

This method is used to expire a session
immediately.

→ if we click on logout or signout links
in a website then internally invalidate() method is

called to expire the session of the client.

→ we can expire the session of a client in
any one of the following two ways.

(1) By setting the expiration time gap b/w request.

(2) By calling invalidate() method.

The following application is to print session information on to browser?

```
import javax.servlet.*;
import javax.servlet.http.*;
public class Session extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
    {
        SessionApp sessionApp = new SessionApp();
        sessionApp.createSession();
        sessionApp.setSessionId(res);
    }
}
```

```
SessionApp
```

```
WEB-INF
```

```
web.xml
```

```
classes
```

```
SessionServ
```

```
Session.java
```

```
*.class
```

```
HttpSession hs = req.getSession(true);  
PrintWriter pw = res.getWriter();  
pw.println("<ch2>");  
pw.println(" session id = " + hs.getId() + "<br>");  
pw.println(" session state = " + hs.isNew() + "  
<br>");  
pw.println(" Inactive Time = " + hs.getMax  
InactiveInterval  
(") + "<br>");  
long ms = hs.getLastAccessTime();  
java.util.Date d1 = new java.util.Date(ms);
```

```
pw.println("u last access time = " +  
          + d1.toString());  
  
pw.println("u fileS");  
close  
pw.println();
```

https://localhost:2011/SessionApp1/session

Session ID = 4E5459EBHED

Session State = TRUE

Inactive Time = 1800

Creation Time = Mon Sep 12 19:32:55 IST 2011

Last Accessed Time = Mon Sep 12 19:32:59 IST 2011

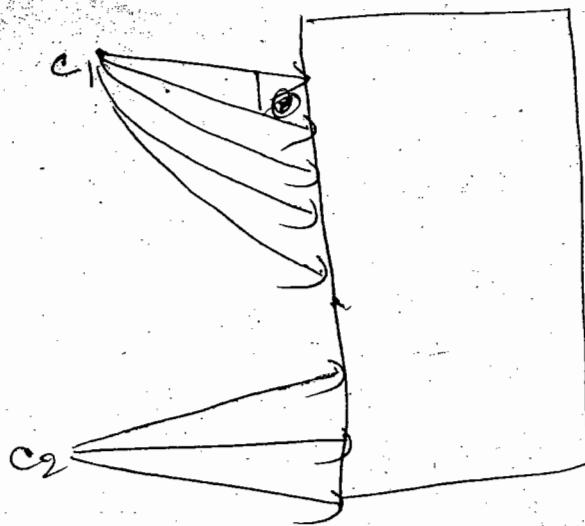
~~Ex:-~~ the following web-application is for counting the number of times request is given by a client to the servlet, by using session of the client.

→ To find individual count of each client, we must use session object, because each client contains a separate session.

→ If the client is new, i.e., if the session is new then start count with 1, & store it into the session object.

→ If the client is existing then, get previous ~~client~~ count value from session & increment it by 1, & again store the new count value back to the session object.

→ If we use Context object in this application, then we will get common count visited by all clients, but we can not get individual count.



5

3

Session APP2

WEB-INF

→ web.xml
classes

CountServlet.java
*.class

CountServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class CountServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req,
                      HttpServletResponse res)
    {
        int count=0;
    }
}
```

HttpSession hs = req.getSession();

if (hs.isNew())

{

count=1;

hs.setAttribute("counts", new Integer(count));

}

else

{

Count++;

hs.getAttribute("counts", new Integer(count));

PrintWriter pw = res.getWriter();

pw.println("<h2> User Visit count = " + count);

pw.close();

web.xml

<web-app>

<Servlet>

<Servlet-name> Session </s-n>

<Servlet-class> CountServlet </s-c>
<Servlet>

<Servlet-mapping>

<Servlet-name> Session </s-n>

<url-pattern> /session </s-n>

</s-n>

<Session-config>

<session-timeout> 1 </session-timeout>

</Session-config>

</web-app>

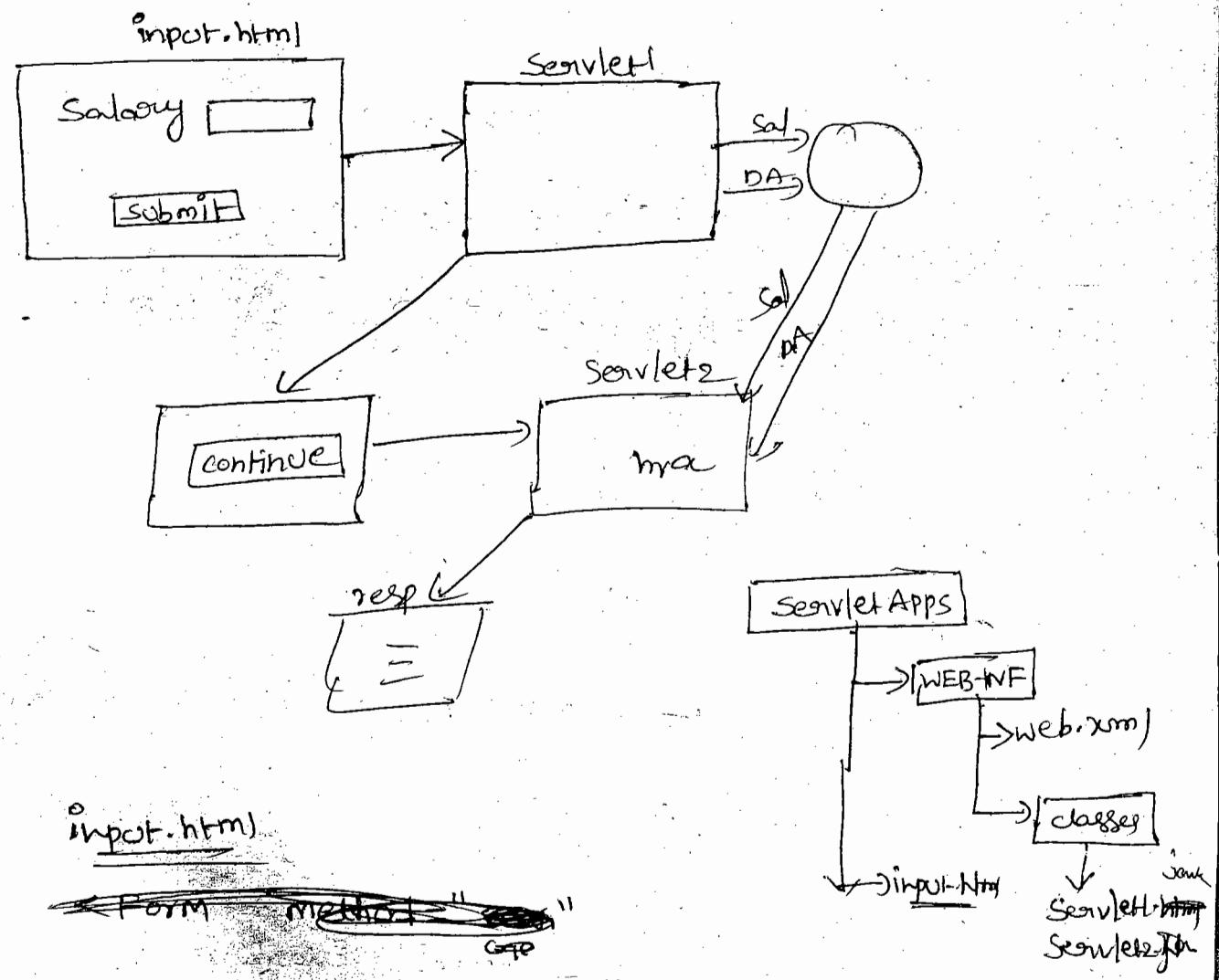
→ when we run the above Example by opening
multiple browser then we will get individual count

So we click on refresh button of the browser after continuous waiting-time of 1 minute then again count starts at one(1). because after 1 minute the previous session of the client will be expired a new session will be started.

Se
=
91

Ex: The following web-application is for using session object & across two servlets visited by the client? {
 In first Servlet, we store Salary & DA values and in second Servlet we read both values from session & we are calculating HRA & printing the response on browser?

pu



type = submit value = click

</form>

Session.java

import javax.servlet.*;

|| || || https://

public class SessionExp3 extends HttpServlet

{

public void doGet(HttpServletRequest req, HttpServletResponse res)

throws SE, IOException

{

String s1 = req.getParameter("s");

int i = Integer.parseInt(s1);

HttpSession hs = req.getSession();

hs.setMaxInactiveInterval(10);

hs.setAttribute("sa", i);

Double dA = i * 0.1;

hs.setAttribute("DA", dA);

}

PrintWriter pw = res.getWriter();

pw.println("<form action='servlet/SessionExp3'>");

pw.println("<input type='submit value="

"Continue">");

pw.println("</form>");

pw.close();

}

```
import javax.servlet.*;
import javax.servlet.http.*;

public class Servlet2 extends HttpServlet
{
    public void doGet(HttpServletRequest req,
                      HttpServletResponse res)
        throws ServletException, IOException
    {
        HttpSession hs = request.getSession();
        Object s1 = hs.getAttribute("Sal");
        String s2 = hs.getAttribute("DA");
        int i = Integer.parseInt(s2);
        Double j = Double.parseDouble(s3);
        Double k = i * 0.20;
        PrintWriter pw = response.getWriter();
        pw.println("Salary is = " + i);
        pw.println("DA is = " + j);
        pw.println("HRA is = " + k);
        pw.close();
    }
}
```

→ In http session technique, container internally generates a cookie b/w transmitting the session ID b/w server & client.

→ Apart from container generated cookie, a servlet programmer can also generate cookies for storing the data of a client.

3) → If a servlet programmer wants to create a cookie then Cookie class of `javax.servlet.http` package is used.

→ Servlet programmers generated cookies are used for state mgmt purpose.

→ With Cookies of Servlet API partial session -

Tracking is possible.

→ When sessionID is not required and when less number of input values are submitted by a client than instead of using HttpSession technique, we can use cookies to reduce the burden on a server.

→ A cookie is an object containing text information in the form of key/value pair and it is created on a server, sent it back to the client and stored on a client browser.

→ In servlet programming, for storing the client information, we need to create cookie object.

→ These objects are created to cookie class.

• `Cookie` - cookie name & value pair obj.

Cookie c = new Cookie(name, value)

String type

- In a servlet, if we create any cookie then container doesn't transfer the cookies from server to browser or viceversa. this work should be done by servlet only.

From a servlet, if we want to transfer cookie back to the browser then we need to add that cookie object to response object, we use

- To add a cookie to the response object, we use addCookie() method.

To read cookies from browser to a servlet, we need to call getCookies method given by request object and it returns an array type of cookie class.

response.addCookie(c1); }
 c1 c2; } }
 " " } Cookies are
 sent from
 server to
 browser

Cookie c[] = request.getCookies(); //cookies

are read from
browser to servlet

Types of cookies

→ 1) in-memory cookie :-

→ By default a cookie is an in-memory cookie.

→ An in-memory cookie lives on a browser until that browser is destroyed (closed).

→ An memory-cookie containing its expiration

browser when the browser is closed & reopened

Persistent cookie :-

If we set some expiration time for the cookie then the cookie is converted from in-memory type to persistent type. In case of persistent cookie, it lives on a browser until its expiration time is reached. It means, even though we close & reopen the browser but still the cookie exists on the browser.

let,

Type

are

some

to

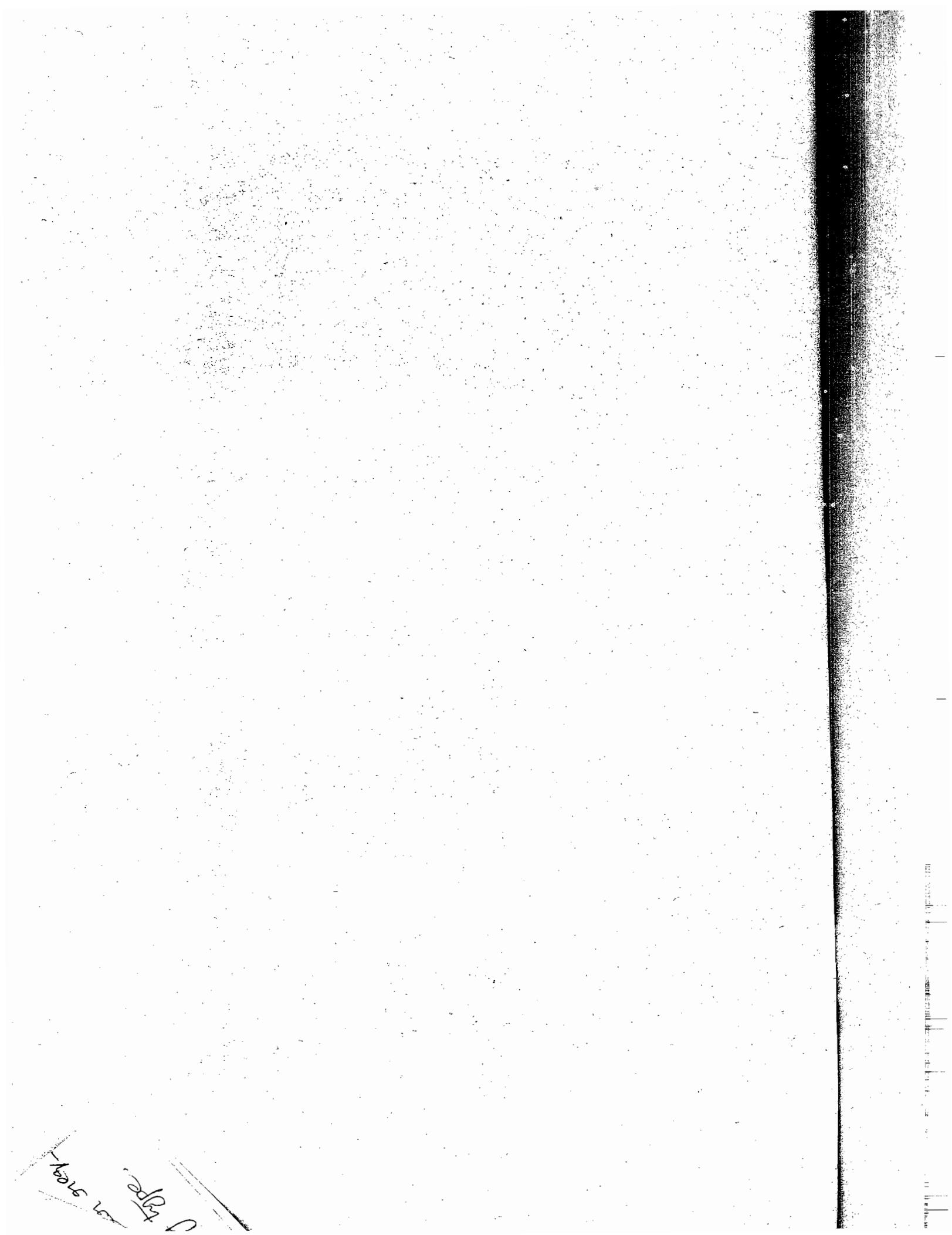
say

like

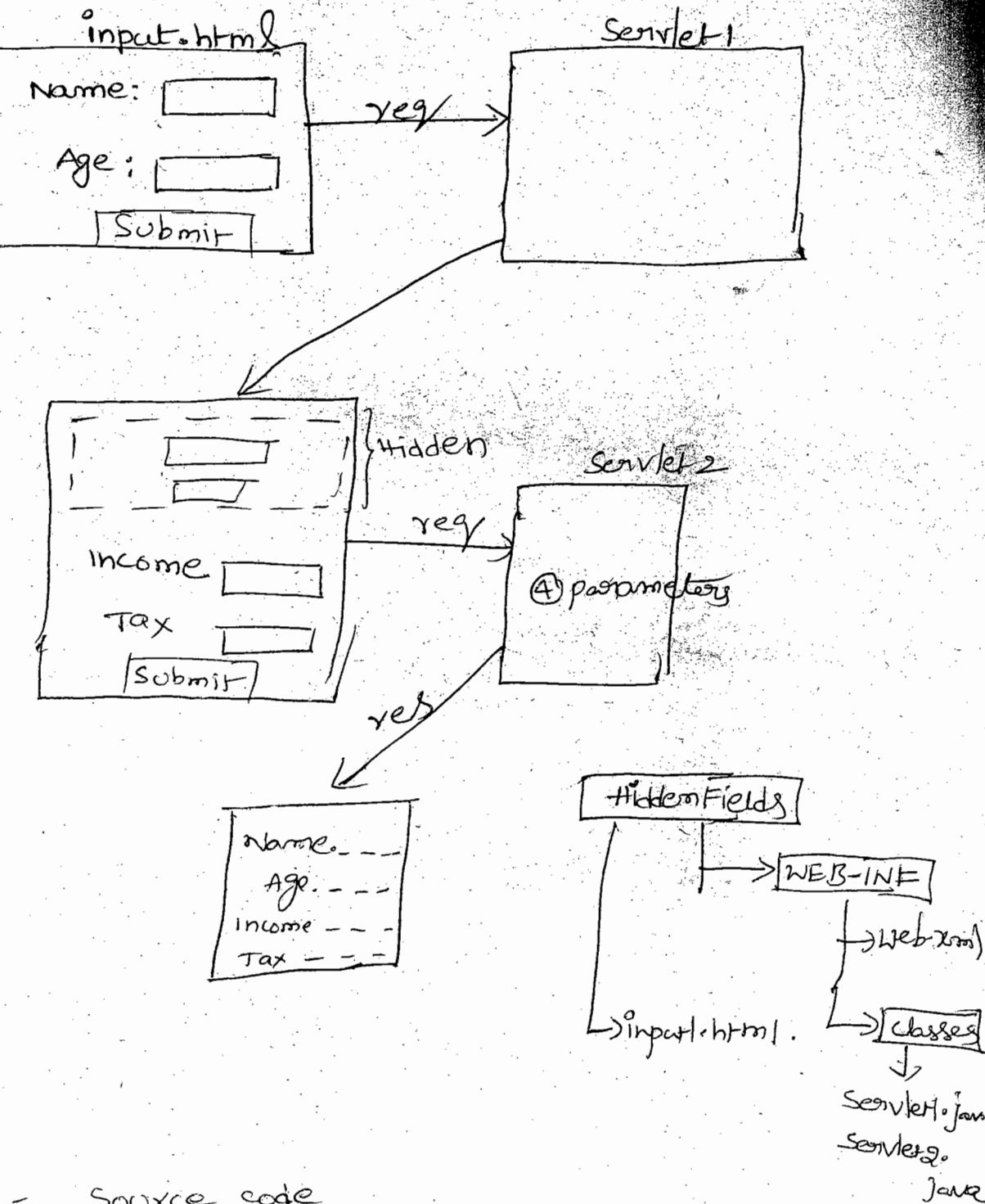
from

convert

'e



16/9/11



Ex:- Source code

page no: 134 , AP-19

Note:

- The hidden fields technique can be applied for both GenericServlet & also for HttpServlet.
- But to use HttpSession & cookies technique we need HttpServlet because in HttpSession & Cookies technique, we need HttpServletRequest &

4. URL - Rewriting

→

→ while working with HttpSession Technique complete session - tracking is possible if it has a major drawback.

to
all

to

→ in HttpSession Session is transferred from server to browser and next from browser to server by storing in a cookie. When cookie is transferred from server to browser, a browser can accept that cookie or even it can reject the cookie. If the cookie is rejected then when next request is given to the server, then along with request a cookie doesn't go to the server.

for

but

or

at

ne

→

nic

but

→ in the above case the container will treat the client as a new client again & again creates a new session-ID & session object for that client. It means the session tracking is not working properly.

[

→

ac

To

→ Finally HttpSession drawback is, if browser accepts cookie then the technique works properly and if browser rejects cookie then the technique becomes failed. In order to overcome the drawback of HttpSession technique,

→

If

.

.

transferred in the form of cookie and also the session-ID will be appended to the URL when next request is going to the server; even though cookie is rejected but session-ID goes to the server from URL so that Container recognizes that this client is an existing client but not as a new client.

→ URL Rewriting is entirely HttpSession based unique only if the sessionID add to the address bar by using a method called encodeURL().

URLRewriting = HttpSession + encodeURL()

→ In URL rewriting mechanism sessionID is added to the address bar with a key called SessionID.

→ To reject cookies from web-site on to Internet Explorer then the following step is required

Tools → Internet Options → Privacy Tab

→ move the settings ticked on to the top (then cookies are blocked) → Apply → OK

How can we say that a web-site uses
URL Rewriting technique or not?

A:- If ~~Session~~ sessionid is printed in the addressbar then we can say that URL-Rewriting technique is used.

Why all Real-time websites uses URL-rewriting mechanism?

As the Real-time applications developed at server-side (or) unknown whether a client browser is accepting (or) rejecting the cookies but sessions should work properly even though cookies are accepted (or) rejected by the browser. For these reason the real-time website uses URL-rewriting mechanism.

Ex:-

The sessionapp3 is now can apply to URL Rewriting we do the following changes in .Servlet.

```
pw.println("<form action = ".+reg.encodeUrl("srv2")+">");
```

Execution:

salary [1000]

[click]

continue

http://localhost:2011/ServletAPP3/Srv1?sal=1000

http://localhost:2011/ServletAPP3/Srv2?jsession

IP = DE0058E6789 EPARBVIE/222

Salary = 1000

HRA = 900

DA = 800

Date
17/9/11

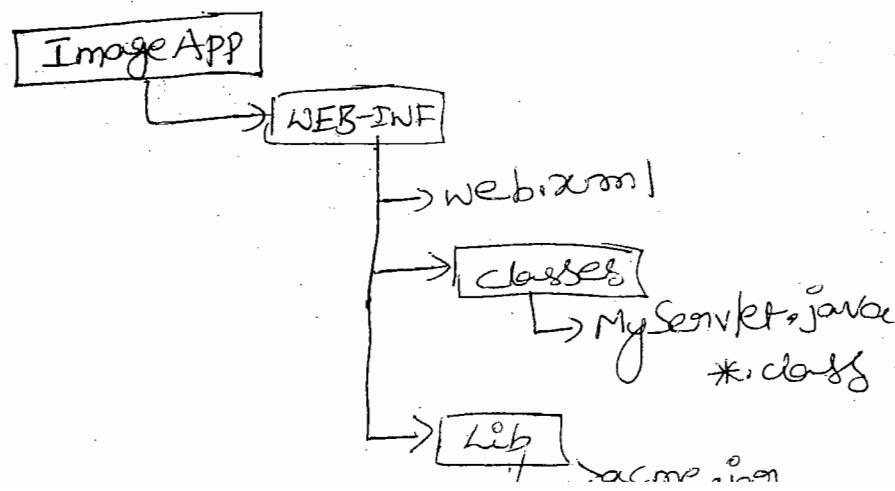
Image Response

→ To send an image as a response from the servlet back to the browser the following changes are required:

- (i) set MIME type (or) Content type as image by using ~~Application~~ ^{image} ~~Content~~ / ~~.gif~~ ^{.gif}
- (ii) To transfer image response, we need ~~servlet output~~ ~~Servlet Output Stream~~ as an output stream to write the response. In this case PrintWriter class is not suitable.
- (iii) use an image encoder class, to store that image file into the output stream object.

Ex:

The following Example is to send an image as a response back to the browser when a request given to the browser.



```
import javax.servlet.http.*;
import java.io.*;
import java.awt.Image;
import javax.swing.ImageIcon;
import Acme.JPM.Encoders.GifEncoder;
```

public class MyServlet extends HttpServlet

```
{ public void doGet(HttpServletRequest request, HttpServletResponse response) }
```

```
response.setContentType("image/gif")
```

```
ServletOutputStream out = response.getOutputStream()
```

```
// obtain ImageIcon
```

```
ImageIcon ImageIcon = new ImageIcon
```

```
("c:/rabbit.gif")
```

```
// obtain image from ImageIcon
```

```
Image image = ImageIcon.getImage()
```

```
// encode the image and send to the output stream
```

```
GifEncoder g = new GifEncoder(image, out)
```

```
g.encode()
```

```
out.close()
```

```
}
```

→ In the above servlet program we have
to use a third Party class called GifEncoder,
so we need a jar file to set in the class
Path called acme.jar

→ to compile the above servlet we need
the following two jar files

- (i) servlet-api.jar
- (ii) acme.jar.

→ we need to copy acme.jar file into lib
folder, in order to execute the web-application
at server-side.

→ copy the image in the specified drive

⇒ Request

http://localhost:2011/ImageApp/srv1

Exception Handling in Servlet

(i) when a client request is given to servlet,
if any exception is occurred at servlet code
then its stacktrace information will be
printed on to the browser.

(ii) The end-user is unknown about
end-user can not

Instead of sending exception on to the browser we can send some error message in a client understandable format back on the browser.

(iv) To do this we need to apply exception handling.

ed by in servlet

(v) In a servlet, exception can be handled in two approaches.

(i) Programmatic approach

(ii) Declarative

→ In programmatic approach, we should include try & catch blocks in the servlet code & if an exception occurred we should send an error page as a response back on the client browser from the catch blocks.

Ex:

P.C TestServlet extends HttpServlet

{
 @Override
 try {
 //
 }
}

=

}

catch (Exception e)

{
 response.sendRedirect("error.html");
}

- In Progammatic approach, we should conclude try & catch block in each servlet individually & it increases burden on the servlet programmer. Instead of handling the exceptions in each servlet separately, we can also handles the exceptions by configuring it into web.xml file we call it tag or declarative approach.
- In case of declarative approach exception handling become global for all servlets & JSP's available in a web-application.
- To configure an exception we should use `<error-page>` tag in the `web.xml`

Ex:-

web.xml

`<web-app>`

— — —
— — —

`<error-page>`

`<exception-type> java.lang.Exception`

`</exception-type>`

`<location> /error.html </location>`

`</error-page>`

— — —
— — —

`<web-app>`

really
not.
see
the
xml

(web-app)

==

<error-pages>

<error-type> java.lang.NullPointerException </error-

<location> /error.html </location> types

</error-pages>

<error-pages>

<error-type> java.lang.ArithmaticException

</error-type>

<location> /error3.html </location>

</error-pages>

— — —

</web-app>

Note: If exception is handled in a servlet
(programmatically) & declaratively (web.xml) then
the preference will be given to programmatic
approach.

→ If we apply Exception-handling in a Servlet
by putting try & catch blocks then it is called
local exception handling & if we handle the
exception through web.xml then it is called
global - exception handling.

JSP

Java Server pages (JSP)

~~~~~ x ~~~~~ x ~~~~~

a) Why JSP?

a) The first technology given for the development of web-applications is Common Gateway Interface (CGI) with CGI technology, there are some drawbacks, so Sun-microsystem released another technology for the development of

web-applications called Servlet.

- ⇒ while working with servlet technology, Sun-microsystem identified some problems
- a) servlet technology need in depth "Java code".
  - b) Servlet Technology is failed to attract the developers working at other technologies into Sun-microsystem technologies side.
- ⇒ In order to get the market, the Microsoft released a web technology called "Active Server page(ASP)"
- ⇒ "ASP" contains a set of predefined tags to develop the web-applications, but it doesn't require in depth coding.

- overcome the problems in Servlet technology in windows or migrated

## to ms- technologies side.

⇒ In order to overcome the drawbacks of Servlet Technology and to make competition b/w ASP, SunMicroSystem released another web-technology called "JSP".

- ① CGI
- ② Servlet
- ③ server pages

a) ASP → MS

b) JSP → SUN

c) PSP → Oracle

Differences b/w Servlet and JSP

| Servlet                                                                                      | → JSP                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ① Servlet need lot of Java code.                                                             | ① JSP need simple tags.                                                                                                                                                  |
| ② In Servlets, if we modify the code, then we need recompilation, reloading and recompiling. | ② In JSP, if we do any modifications, then just we need to click on recompile button and recompiling, reloading are not required, so it is not a time-consuming process. |

③ In Servlet, we need to implement business logic, presentation logic combined. It is not possible to separate business logic & presentation logic. we call these problems as a page centric problem.

④ In Servlet we do not have implicit objects. It means, if we want to use an object then we need to get object explicitly from the server.

⑤ In Servlet, by default session mgmt is not enabled. we need to enable explicitly

⑥ Servlet accepts all protocols request

⑦ In a Servlet, all packages must be import on the top of servlet.

③ In JSP, we can separate them business logic from the presentation logic, by using Java Beans Technology. so page centric problem is eliminated.

④ In JSP, we have implicit object Support.

⑤ In JSP Session mgmt is automatically enabled.

⑥ JSP will only accept http protocol request.

⑦ In JSP packages can be import anywhere i.e., top, bottom or middle.

⑧ In servlet, service() need to be overridden

⑨ In Jsp we —  
override - jsp service

⑩ In servlet, we have only three scopes i.e.,  
request  
session  
Application

⑩ In Jsp, we have four scopes.  
page  
request  
session  
Application.

⑪ servlet only supports dynamic including

⑪ jsp supports both static and dynamic including

⑫ Servlet init()  
we can use parameters.

⑫ Jsp init() does not use the parameters.

Q) what is a Jsp?

Jsp is another web technology given by "SUN" for the development of dynamic web pages on to the client browser.

⇒ A Jsp is called as page but not a program, because a Jsp contains totally tags.

⇒ A servlet is a program, we insert "complex business code".

⇒ Jsp or page writer is called as "author" and a servlet programmer is called as "developer".

⇒ Every Jsp is internally converted into a servlet and by the server (container).

Jsp Tags

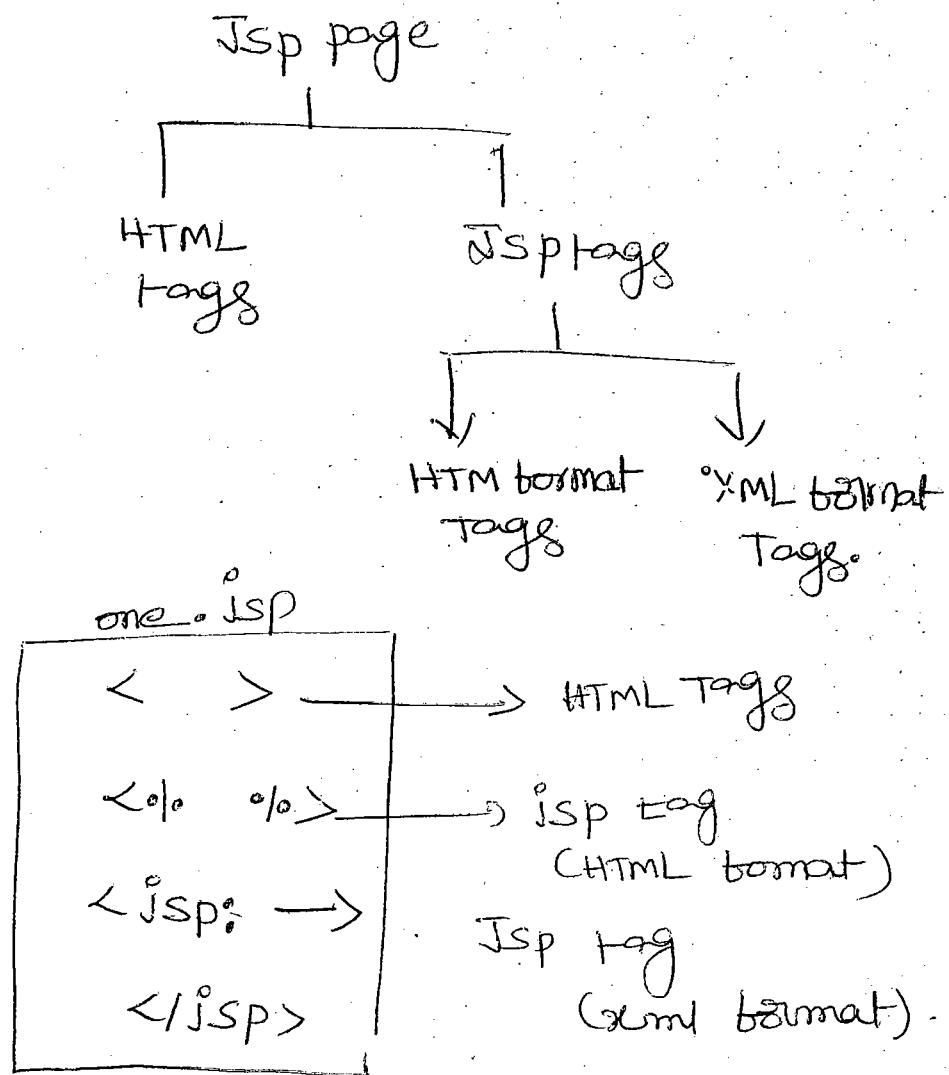
~~~~~

→ A Jsp page contains both "html tags" and "Jsp tags".

→ HTML tags will produce "static output" and Jsp tags will produce "dynamic output".

→ Because of Jsp tags we call a Jsp tag a

"dynamic web page".



Jsp Translation :-

⇒ Each Jsp page is internally translated into an equivalent Servlet.

⇒ In a Jsp page construction, the programmer writes some html tags and most of the Jsp tags inside it.

⇒ Every Jsp tag written in the page will be internally converted to an equivalent Java code.

by the containers. we call this process as "Translation".

⇒ In Jsp runtime, there are two phases

1) Translation phase.

2) Request Processing phase.

⇒ Translation phase means converting a Jsp into an equivalent servlet and then generating class file of the servlet.

⇒ Request processing phase means, executing service method of Jsp and generating response data on to the client browser.

⇒ when first time request given to a Jsp then translation phase occurs first and then after service phase will be executed.

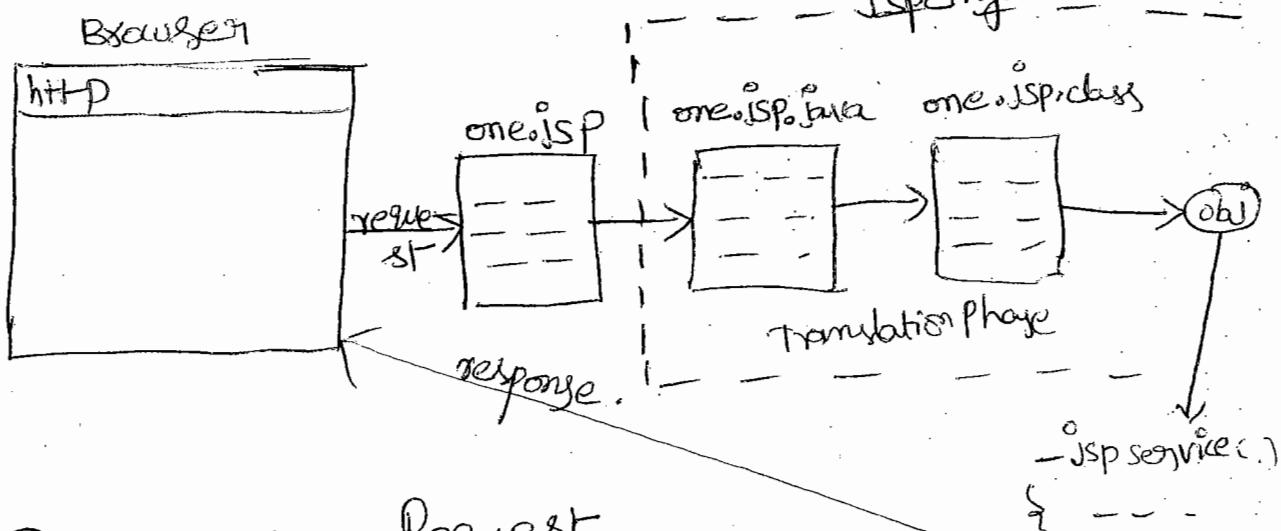
⇒ From next request from the Jsp, only request processing phase execution but translation phase is not executed because Jsp is already translated.

⇒ If a request is given to the Jsp after it has modified then again translation phase is executed and after that request processing phase/service

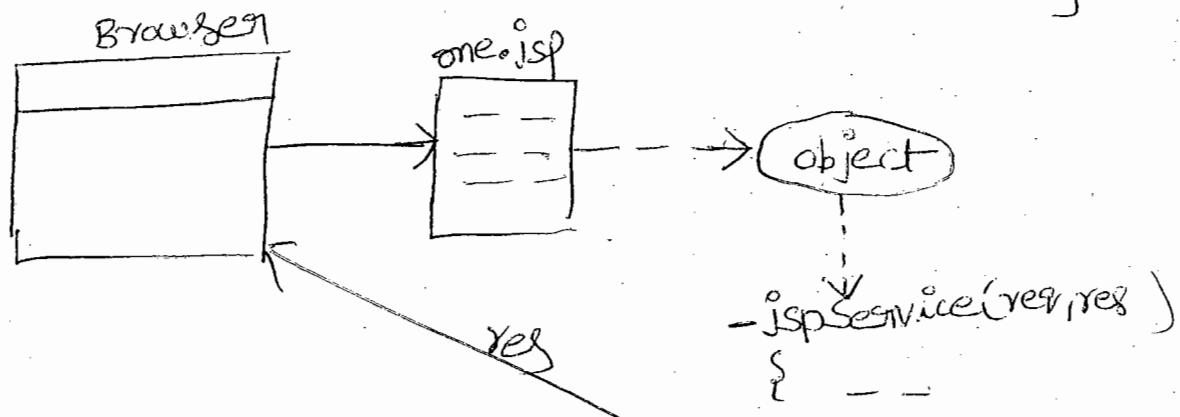
- ⇒ In the following
- when first request is given to the Jsp.
 - when a jsp is modified.

Jsp Translation

when first request or when Jsp is modified



② From Next Request



Q) How the web container will recognize when a jsp is modified or not?

A) Jsp container will use file compression tools like Araxis to verify whether a jsp is modified or not.

→ file compression tools internally uses timestamp information to check whether modified

cations are done on a file & not.

Note:

- 1) If we shutdown & restart the server then only requestProcessing phase will be executed. because, when we shutdown the server, only servlet object is destroyed but the servlet java and class files are not deleted from the server.

- 2) If we remove the class file from the server then partial translation occurs.

Life cycle - methods of JSP

- ① `jspInit();`
- ② `-jspService(req, res)`
- ③ `jspDestroy();`

→ As a JSP programmer, we can override `jspInit()` and `jspDestroy()`, but we cannot override `-jspService()` to inform the programmer that we should not override. we inform the service() method name is started "underscore" symbol.

⇒ when a JSP is translated into an equivalent servlet then that servlet class extends some base class provided by the server but "indirectly".

④

`<%>`

`a+b, c+d;`

`%>`

⑤ `<% = a, b %>` → Invalid, becoz it is equal to `out.println(a, b);`
we can't point the variable by separating width.

⑥ `<% = a+b; %>` → Invalid, because it is equal to `out.println(a+b);`

⑦ `<% = request.getParameter("uname") %>` → valid.

In this expression we have used an implicit object `request`, This expression print "uname"

Request parameter value on to the Browser

In JSP, the implicit objects are allowed into the tags, which are inserted into `JspService` method.

An expression tag - `JspService` method only.

So implicit objects are allowed in the expression tag.

In declaration tag, implicit objects are not allowed because declaration tag goes to outside of -IsService Method.

Ex:-

<%!

String s = request.getParameter("uname") %>



invalid

Ex:-

one.jsp

<%! ①
private int a=10, b=20;
private String s="satya" %>

→ inserted outside
the
service
method

→ Declaration
tag
→ Expression
tag

②
<% = a+b %>
<% = s %>
<% = a+ " " +b %>

→
→ inside
the
service
method

one-jsp.java

public class one-jsp extends HttpServlet

{

①

private int a=10, b=20;

private String s="string";

public void-jspService(req, res) throws SE, IOE

{

≡

②

out.print(a+b)

" " "(s);

" " "(a+" "+b);

≡

op

30

satya

10 20

-JSF

③ Scriptlet tag:-

These tag is used to insert java code in a JSP. even though there are multiple tags given in JSP but still we need to insert some small piece of Java code also in a JSP.

Scriptlet tag is used for implementing the business logic in a JSP.

Scriptlet tag goes to JSPService(), during the translation. It means Scriptlet tags are executed for each request.

<%>

Java code

<%>

Ex:-

<%>

String s1 = request.getParameter("uname");

String s2 = request.getParameter("pwd");

if (s1.equals(s2))

out.println("ok");

else

out.println("Sorry");

{
JSPservice() goes to }

- we can use implicit objects of the JSP in a scriptlet tag also, because scriptlet tag goes to - isService() only.
- In a JSP Implicit objects are allowed in expression tag & scriptlet tag but not in the declaration tag.

Ex:-

<%!
int a=100;
%>

A bracket on the right side groups the entire code block from the start symbol to the end symbol, labeled "Declaration Tag".

<%
int a=49;
%>

A bracket on the right side groups the entire code block from the start symbol to the end symbol, labeled "Scriptlet tag".

<% a %>

A bracket on the right side groups the entire code block from the start symbol to the end symbol, labeled "Expression tag".

<% = this.a %>

A bracket on the right side groups the entire code block from the start symbol to the end symbol, labeled "Expression tag".

⇒ In the above Example we can create variable or in both declaration tag & scriptlet Tag ✓

Jsp
et

⇒ In this declaration,

declaration tag variable → global variable

scriptlet tag variable → local variable.

Expression tag also insert inside the -^(service)
so it give the preference to local variable
i.e., scriptlet tag variable. if we want to
print global variable we refer it by using
'this' keyword.

③

L10

```
public void m1()
{
    }
%>
```

} Invalid.

⇒ ④ Invalid. Because the scriptlet tag goes to
-ispService method. In java nested method defini-
tion are not allowed.

In a scriptlet we can declare variables

but not methods. because variables will be come

④

<%>

int a=40;

} Invalid.

<% = a %>

<%>

Invalid. because we can not write one scripting element in another scripting element.

Ex:

one.jsp

<%>

int a=100, b=200;

%>

<%>

int c=0;

c=a+b;

%>

<% = c %>

Expression Tag.

one-JSP.java

public one-JSP extends HttpServlet

Body

int a=100;

int b=200;

public void JSPService(HttpServletRequest request, HttpServletResponse response)

throws ServletException

int c=0;

c=a+b;

out.println(c);

}

⇒ the scripting elements of JSP can we
written either in HTML Syntax or in XML

Syntax

①	HTML Syntax	XML Syntax
I	<u>Declaration tag</u> <code><%!</code> <code>int a=200, b=100;</code> <code>%></code>	<u>I Declaration tag</u> <code><jsp:declaration></code> <code>int a=200, b=100;</code> <code></jsp:declaration></code>
II	<u>Expression tag</u> <code><% =</code> <code>expression</code> <code>%></code>	<u>II Expression tag</u> <code><jsp:expression></code> <code>expression</code> <code></jsp:expression></code>
III	<u>Scriptlet Tag</u> <code><%</code> <code>java code</code> <code>%></code>	<jsp:scriptlet> <code>java code</code> <code><jsp:scriptlet></code>

Jsp comments

~~~~~

(i) Jsp Comment | Hidden comment

(ii) HTML comment

(iii) Java comment

Jsp comment

~~~~~

This type of comment is called as hidden comment because it is invisible when a jsp is translated into a servlet internally.

Sym :-

<%-- Comment --%>

Ex:-

<%-- one.jsp --%>

(ii)

HTML comment

If it is a comment tag for HTML, XML and Jsp.

In a JSP page, if we write HTML Comments, then these comments are visible - `JspService()` method of the internal servlet.

Synt:-

`<!-- comment -->`

Ex:-

`<!-- one.html -->`

`<!-- web.xml -->`

(3) Java comment

we can write either a single or multiple line comments of Java in a Scriptlet tag of the JSP. Java Comments are allowed only inside Scriptlet tag because we can insert Java code in JSP only at Scriptlet tag.

Synt:

`// Single Comment`

`/* --- */ multi-line.`

Ex:-

`<%`

`// if condition`

Configuring a JSP in web.xml

Configuring a JSP into web.xml file is optional. because JSP is a public-file of the web-application.

A JSP is called a public file, & a servlet is called a private file of the web-application because JSP file stored in root directory of the web application, & servlet class file is stored in subdirectory of the web application.

Because JSP is a public file, we can directly send a request from a browser by using its filename.

If we want to configure a JSP in web.xml file then the XML elements are same as servlet configuration. we need to replace servlet-class

element with jsp-file element

Ex:-

web.xml

<web-app>

<Servlet>

<Servlet-Name> test </S-N>

<jsp-file> /one.jsp </jsp-file>

</Servlet>

<Servlet-Mapping>

<s-n> test </s-n>

<url-pattern> /s8 </url-pattern>

</S-m>

</Web-app>

If we configure a jsp in web.xml then
we can send request to the jsp either
by using jsp-file name or by using its
<url-pattern>.

http://localhost:2011/root/one.jsp

http://localhost:2011/root/s8,

Ex:10

The following JSP application is for
Counting the number of Requests given to
the JSP.

Ex:- one.jsp

<%-- one.jsp --%>

<%!

int count=0;

%>

<%

count++;

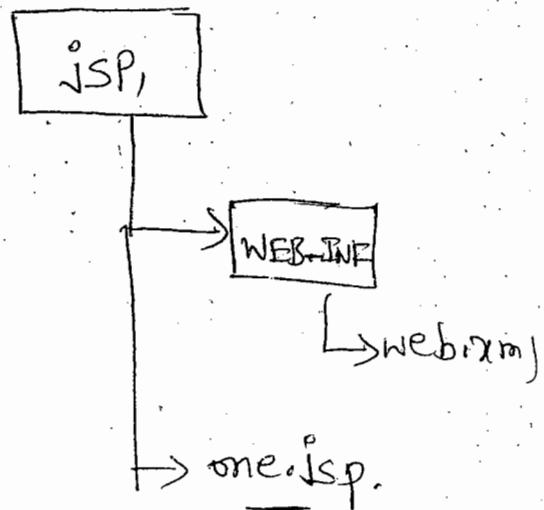
%>

This ~~is~~ is the request; <%= count %>

=> web.xml

<web-app>

</web-app>



Q1) $\text{http://localhost:2011/JSP1/one.jsp}$

This is request:1.

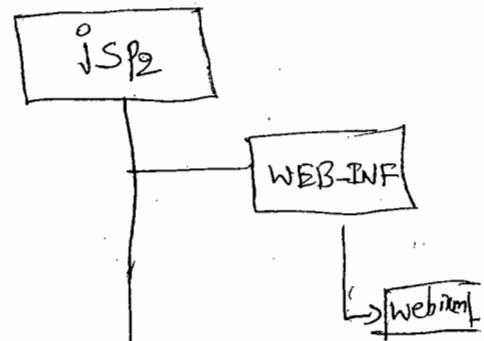
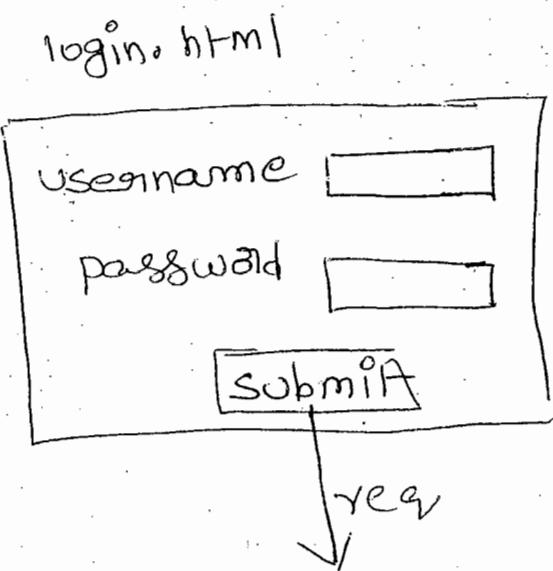
The Servlet class generated by the container
is available at the following location.

C:\Tomcat 6.0\work\Catalina\localhost\JSP1\org\

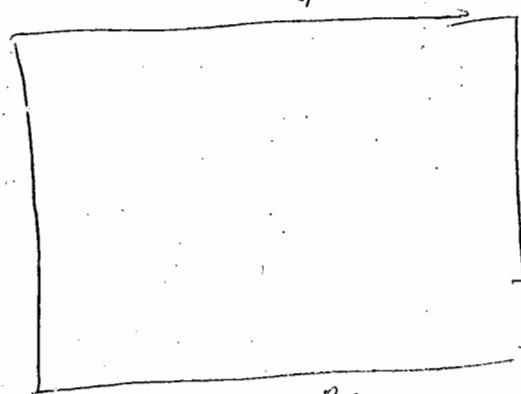
apache\jsp
pkg folder

Ex:-2

The following example is for communicating
from HTML to JSP for a login app?



→ login.html
→ login.jsp



req

Login Success
Failure

Login.html

<!-- Login.html -->

<form action="Login.jsp">

username: <input type="text" name="t1"> >

password: <input type="password" name="t2" /> >

<input type="submit" value="click" />

</form>

Login.jsp

<% -- login.jsp -->

<%

String s1 = request.getParameter("t1");

String s2 = " " ("t2");

int k = Verify(s1, s2);

if (k == 1)

out.println("login success--");

%>

<%! int verify (String p1, String q1)

{

if (p. equals ("satya") && p. equals
 ("satya"))

return 1;

else

return 0;

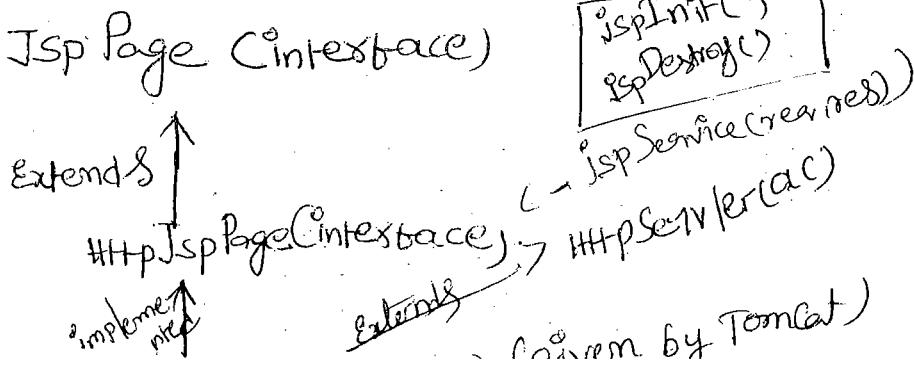
}

%>

⇒ If jsp is translated after giving the first request,

<load-on-startup> 1 </load-on-startup>

- ① Jsp life - cycle are provided by two interfaces
- the Jsp - API
 - (i) javax. servlet. jsp. JspPage
 - (ii) javax. servlet. jsp. HttpServletJspPage
- Jsp page interface has provided two life - cycle methods
- (i) Jsp. init()
 - (ii) JspDestroy()
- HttpServlet page interface has provided a single life - cycle methods called - JspService (req, resp).
- In case of Tomcat Server the Tomcat container Extends the internal Servlet from HttpServletBase class.
- HttpServletBase is an abstract class Extends from HttpServlet and implemented from HttpServletPage interface.



- aceg
- In `HttpJspBase` class given by tomcat, there is one abstract method. so the class is given as an abstract class.
 - In `HttpJspBase` class, both Servlet life-cycle methods and JSP life-cycles are overridden and JSP cycle methods are called from Servlet life-cycle methods.
 - A webContainer can only call Servlet life cycle but not JSP life-cycle. so container calls Servlet life-cycle and Servlet-life cycle calls JSP life-cycle.
- Ex:-
- ```

public abstract class HttpJspBase extends HttpServlet
 implements JspPage
 {
 public void init(ServletConfig config) throws ServletException, IOException
 {
 jspInit();
 }
 public void service(HttpServletRequest req, HttpServletResponse res)
 {
 jspService(req, res);
 }
 public void destroy()
 {
 jspDestroy();
 }
 public void jspDestroy()
 {
 jspService(req, res);
 }
 }
}

```
- extended  
by  
HttpJSP

class

```

public abstract void jspService(HttpServletRequest req, HttpServletResponse res)

```

Note → For every Servlet class internally created by the "Container", the base class is "HttpServlet indirectly".

"Implementation" in the JSP package

## Jsp Elements

- ① Scripting elements
- ② Directive elements
- ③ Action elements
- ④ custom elements.

### Scripting elements

- ① Declarations
- ② Expression
- ③ Scriptlet.

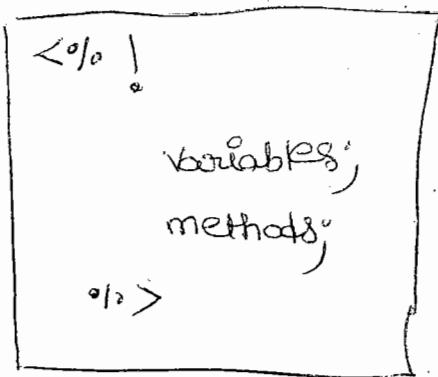
Declaration:-

It is used to create variable & methods in a JSP.

→ The variable and methods will become to global to that JSP. It means in that JSP we can use those variables anywhere and we can call those methods anywhere.

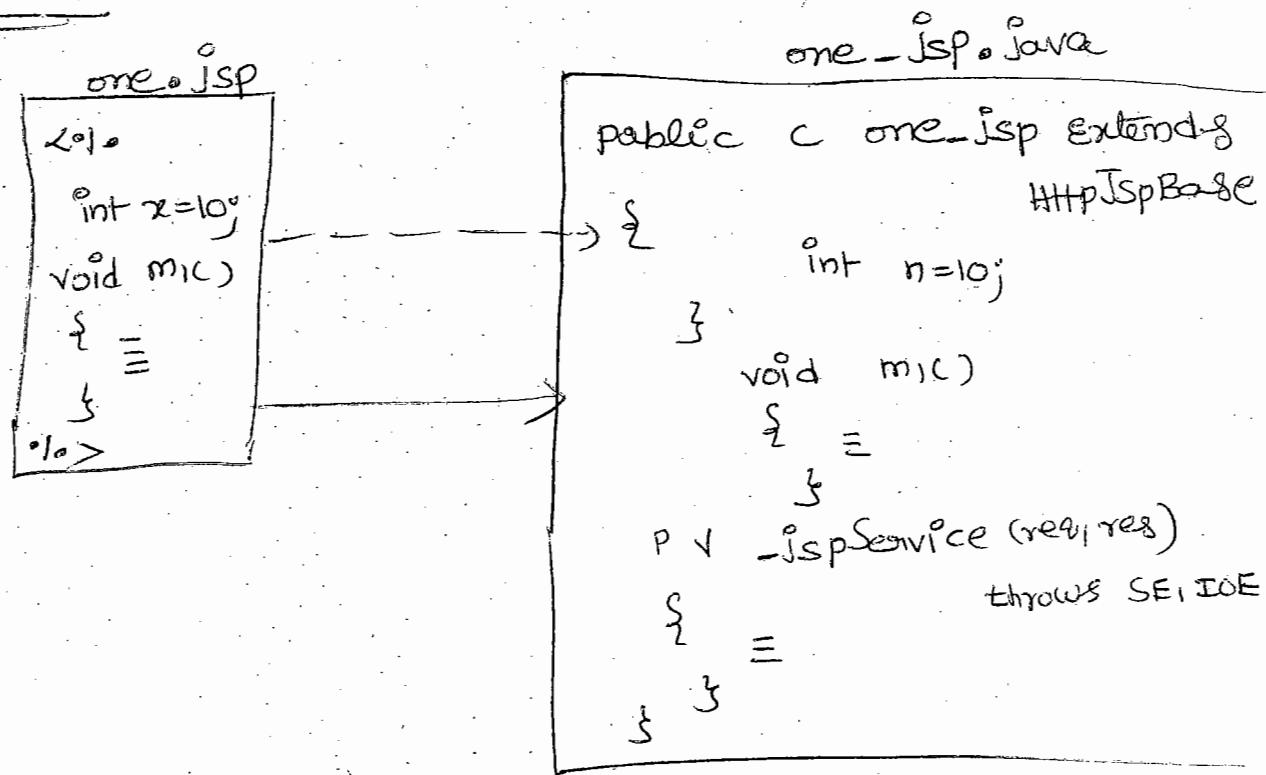
→ At the time of translation, container inserts the declaration tag into the class. So the variables will become instance variables and methods will become instance methods.

→ Syntax:



Example:

next



## ② Expressions

(1)  $\langle \text{!} = \begin{cases} \text{converted} \\ a+b \end{cases} \rangle \rightarrow \text{out.print}(a+b) \text{ goes to } -\text{jspService}(\text{req}, \text{res})$

(2)  $\langle \text{!} = \begin{cases} \text{invalid, becos of two expressions} \\ a+b \\ c+d \end{cases} \rangle$

(3)  $\langle \text{!} = \begin{cases} \text{valid } \rightarrow \text{out.print} \\ a+b + " " + c+d \end{cases} \rangle$

Adding client-side validation to the ex.

- If you don't apply validation on client side to the input then even though the input is not given, the request will be submitted to the server.
- If a request is submitted to the server by without validating the input at client side than the no of round trips b/w client & server will be increased. So before we submit the request to server, first we need to verify whether the client input is entered or not. This is possible by enabling the client-side validation through JavaScript.
- Before submitting the request to server, the first JavaScript function will be executed and if that fn returns true then the request will be submitted to the server and otherwise the request will not be submitted.

⇒ The Advantage of client-side validation is, we can reduce the no of round trips b/w client and a server.

⇒ To add a JavaScript function, to the HTML or JSP page then we should add the script tag under head section and function calling should be done from the body section

⇒ To call javascript function, whenever submit is clicked then we need to add a javascript event handler to the <form> tag called on Submit

```
<form action = "Arithmetic.jsp" name = "f1" onsubmit = "return fun1(this)">
```

```
<! --- input.html --- >
```

>

```
<html>
```

```
<head>
```

```
<script language = "JavaScript">
```

```
function fun1(form)
```

```
{
```

```
var x = document.f1.t1.value;
```

```
alert(x);
```

```
var y = document.f1.t2.value;
```

```
if (x == "" || isNaN(x))
```

```
{ alert("A value is not a number"); }
```

```
else if (y == "" || isNaN(y))
```

```
{
```

```
 alert ("B value is not a number");
```

```
 return false;
```

```
}
```

```
else
```

```
 return true;
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<center>
```

```
<h2>
```

```
<form name = "f1" action = "Arithmetic.jsp"
```

```
onSubmit = "return fun1 (this)">
```

```
Enter a: <input type = text name = "t1">

```

```
Enter b: <input type = text name = "t2">

```

```
<input type = submit value = "Add"
```

## JSP Directives

⇒ Directives • doesn't produce any output on to the browser directly.

⇒ Directives are used to add some additional behaviour to the Servlet, during translating a JSP into a Servlet.

⇒ In JSP we have the following three directives:

- ① Include ② page ③ Taglib.

⇒ A directive starts with `<%@` and ends with `%>`

`<%@ directive_name attribute %>`

- ① Include

→ This include directive is used to include the code of one JSP into another JSP during translation time.

→ If we include one JSP into another JSP then during translation time, the JSP engine creates a single servlet internally.

→ If we write include directive then code of one JSP is included to another JSP, but not the response at runtime, so we call the include directive as static including (at compile time including) or inline including.

→ In a single JSP page, we can include any no of other JSPs. It means we don't have any restriction.

→ While including, both source and destination pages must be JSP pages only.

Ex:- f1.jsp

=

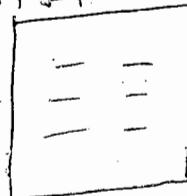
<%@ include file = "f2.jsp" %>

=

<%@ include file = "f3.jsp" %>

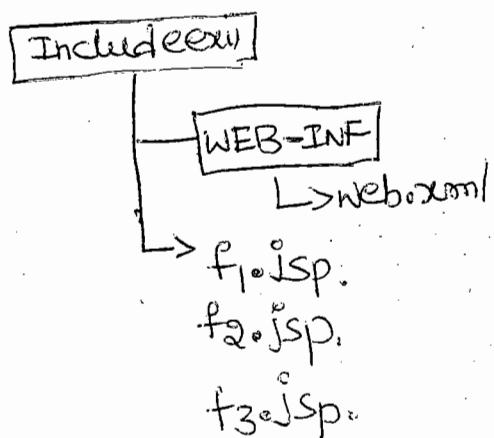
f1.jsp.java

f2.jsp  
---  
---  
---  
---  
---  
f3.jsp  
---  
---  
---  
---  
---



Note:-

In include directive, only one attribute and that is "file".



## f1.jsp

<%-- --%>

<%>

out.println("Hello");

<%>

<h>

<%@include file="f2.jsp"%>

<b>

<%@include file="f3.jsp"%>

## f2.jsp

<%-- --%>

<%>

out.println("Good morning");

<%>

## f3.jsp

<%-- --%>

<% Date() %>

Request:-

http://localhost:2001/includeex/f1.jsp

\*\* note: include directive reduces the burden on a server, because by multiple JSPs internally, only one servlet is created so one servlet only one object is required. It means no of servlets objects are reduced.

## ② Page Directives

⇒ This page directive is for setting important behaviour to a servlet during the translation.

⇒ The important behaviour means like exception handling, session mgmt, packages importing, mime type settings etc.,

⇒ Syn:

<%@ page attribute %>

Attribute:

(i) Language: -

→ The attribute is given to provide the language supported by the "JSP tags".

→ The default language and the only supported language is "Java".

[<%@ page language = "Java" %>]

(ii)

Import: -

→ These attribute is used to import- either predefined or userdefined packages into a JSP.

⇒ while importing the packages into a JSP, we can import anywhere in the JSP. It means at the beginning of JSP or in the middle or at the end.

⇒ we can import multiple packages also at a time by separating package name with either with a comma, semicolon.

⇒ Even though we can import the packages at multiple places in the JSP, but during translation all import are placed on top of the Servlet.

⇒ The only one attribute of page directive, that can be "repeated with different values" is import attribute.

Ex:-

<%@ page import="java.util.\*; java.sql.\*" %>

<%@ page import="java.net.\*" %>

<%@ page import="java.text.\*" %>

③ session:

⇒ For Example-

(i) one.jsp

<%@ page session="false" %>

} wrong

<%@ page session="true" %>

ex:-

(ii)

<%@ page session="false" %>

### Ex:-3 one.jsp

<%@ page session="false" session="true" %> } → wrong

<%@ page session="true" session="true" %> } → correct  
demo.jsp

<%@ -- demo.jsp -- %>

<%@ page import = "java.util.\*" session="true" %>

<h3>

The current date & time is: <% = new Date() %> <br>

Session Id : <% = Session.getId() %> <br>

Inactive time : <% = Session.getMaxInactiveInterval()  
() %> <br>

Session state (new /old) : <% = Session.isNew  
() %> <br>

</h3>

### 4) Error page

→ while executing a jsp, if any exception is occurred then that exception stackTrace will be printed on the client browser.

→ If we want to transfer the control to another jsp whenever an exception is occurred in the current jsp then we need to add error page attribute of the page directive.

→ If we add error page to a jsp then the control is

transferred to the error page, only if the exception is occurred. otherwise the control will not be transferred.  
→ If we add error page attribute to the page directive in a JSP then we call it as "local" Exception handling

### 5) isErrorPage

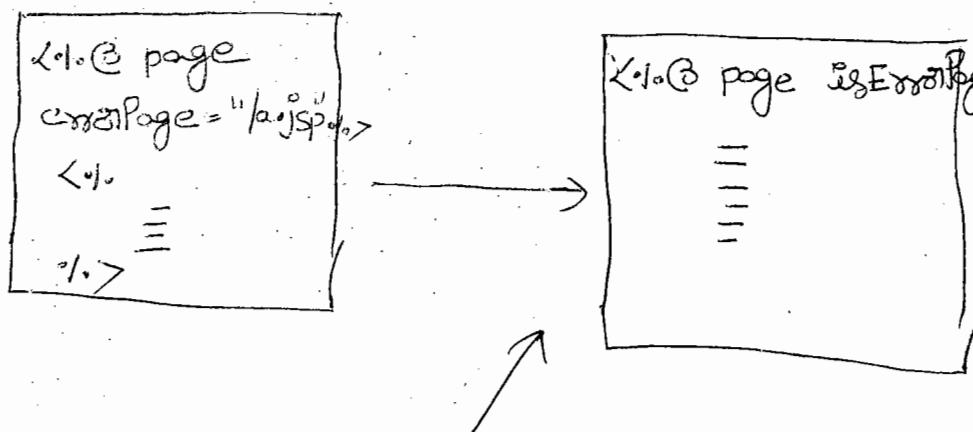
→ This attribute is used to make a JSP as an error page.

→ The default value of this attribute is false. It means by default a JSP is not acting as an error page.

→ If we write `isErrorPage="true"` then that JSP is acting as an errorpage.

→ If a JSP is acting as errorpage then only an

implicitly object called "exception" is available in that JSP page. otherwise we cannot access any user exception object in the JSP page.



red  
on  
JSP  
ISP  
he  
14

## 6) isThreadSafe:-

- The default value of this attribute is "true".
- By depending on the value of isThreadSafe, a container allows either multiple threads (or) single thread to access the JSP page.
- If we write isThreadSafe = "false" then container allows single thread at a time to access the JSP page.
- If isThreadSafe = "true", then container allows multiple threads to access the JSP at a time.
- If we write isThreadSafe = "false" then the internal servlet created by the container implements "singleThreadedModel" interface.

## 7) buffer:-

- the default value of this attribute is "8KB".
- we can increase (or) decrease the buffer size by using the buffer attribute.
- In JSP, we use an implicit object for generating response called "out".

true  
be,  
)  
o  
288  
allows  
D  
oney  
)  
18kb-  
ize  
ating

⇒ "out" is an object of "JspWriter" class and it makes use of buffer internally.

⇒ JspWriter is similar to PrintWriter but the difference is PrintWriter doesn't use buffer. but Jsp writer makes use of "buffer".

JspWriter = PrintWriter + Buffer

Ex:

pw.println(" ") → response  
pw.println(" ") →  
pw.println(" ") →

out.println → Buffer  
out.println → allocation  
out.println → Yes

First time generated Instruction stored in JITCompiler

JspWriter = PrintWriter

⇒ A buffer is like a cache memory and at first time an application executes them it makes some slow performance. but when next time onwards it increases the performance of an application.

⇒ In a Jsp page we can do the buffer settings by using buffer attribute

- ⇒ if we want to make up  
we should make buffer as name
- ⇒ automatically the dat will be opened into response obj  
from a buffer by internally calling flush method.
- ⇒ we can also explicit call the flush method by using  
autoflush statement.

### AutoFlush

```
<%@ page buffer=10kb autoflush="true" %> (
```

- this attribute is used to set flush mode in  
Jsp flushing means synchronizing the data from  
buffer memory to response object
- the default value of autoFlush is True
- If autoFlush mode is disable to writing auto-  
flush is equal to false then in a Jsp page  
the programmer has to call the flush method explicitly
- If buffer is remove than the programmer  
has to set autoFlush is equal to False otherwise  
Jsp Container throwing an exception

Extends:-  
⇒ this attribute is used to provide the class name from  
which we want to extends internal servlet class.

- when a Jsp is converted into equivalent servlet  
then the internal servlet class extends some base  
class which is provided by the container instead  
of extending from the container given class, we can

also specify our class name by writing extends

attribute

one.jsp

Like page extends "satya" :->

public class one.jsp extends satya

{

=

}

> If we want to specify our class name then our class should extend the server given class or our class should extend HttpServlet and it should implement HttpServletRequest interface.

Should implement HttpServletRequest the class satya

→ In case of tomcat server the class satya should extend HttpServlet class or the class satya should extend HttpServlet and it should implement HttpServletRequest interface.

public class satya extends HttpServlet

{

=

}

(a)

public class satya extends HttpServlet implements

{

=

}

HttpServletRequest

ing

>

m

m

auto-

re-

explici-

707

class-

net-

page-

end

## Content Type:

① `<@. @ page.ContentType = "text/html" %>`

(i)

`<@. @ response.setContentType("text/plain") %>`

(ii)

→ These attribute is used to set mime type, the default mime type is `text/html`.

→ If we want to set mime type in a JSP then we have two ways-

(i) We can call `setContentType()` if we can

`use ContentType = "text/html" >`

Info

`<@. @ page.info = "this is a sample page" %>`

→ This attribute used to write some description for a JSP page, we can also write description by using Comment Tags but it is not possible to read the description written in the Comment tags

→ In case of Info attribute we can need the comment one description whenever it is required by calling a method `getServletInfo`

<%= getServiceInfo() %>

(81)

<% String s = getServiceInfo();

out.println(s);

%>

isELIgnored

⇒ EL means expression language

if by default EL is not Executed in JSP

<%@ page isELIgnored="false" %> → Enable

⇒ In JSP by default EL is disabled: it means  
EL will not be executed -

the default value of isELIgnored = "true", if

we make this attribute as false then expression language will be Executed.

XML:

<jsp:directive include file = "twoJSP.jsp" />

<jsp:directive page import session = 1>

Note:

\* JSP directive can be written using XML  
syntax also

<jsp:directive page "import = "java.util.\*" />

⇒ standard actions is used for predefined tags.

⇒ custom tags are used for userdefined tags.

⇒ standard actions base for what and

what logics are used in presentation  
and business logic.

⇒ xml tags are used for standard actions

⇒ we can tag is used for "object" and  
check if object of the id is reference of  
object.

### standard actions

→ standard actions are given in JSP to separate  
the presentation logic and the business logic  
of the JSP.

→ The name is given as a standard action,  
because each action has a predefined meaning  
and as a programmer it is not possible to  
change the meaning of the tag.

→ standard actions are not completely separating the

presentation logic with the business logic, but  
partial separation of the two logic is possible.

→ each standard action follows "xml" syntax. We do

not have any "equivalent" XML syntax.

→ The standard actions given by JSP are

- 1) <jsp:useBean> 2) <jsp:setProperty> 3) <jsp:getProperty>
- 4) <jsp:forward> 5) <jsp:includes> 6) <jsp:params>
- 7) <jsp:parameters> 8) <jsp:plugin> 9) <jsp:tagbody>

## 1) <jsp:useBean>

→ In JSP, it is possible to separate the presentation logic from its business logic

→ If we want to separate both presentation logic and business logic in JSP, we need to create a Java bean.

→ We can implement the business logic into a JavaBean and we can call that business logic from a JSP.

→ The advantage of separating business logic into a JavaBean class is, we will get reusability. It means that JavaBeans business logic can be called from any JSP.

→ In a JSP, if we want to call the business method of the JavaBean class, then first of all we need an object of the JavaBean class into a JSP.

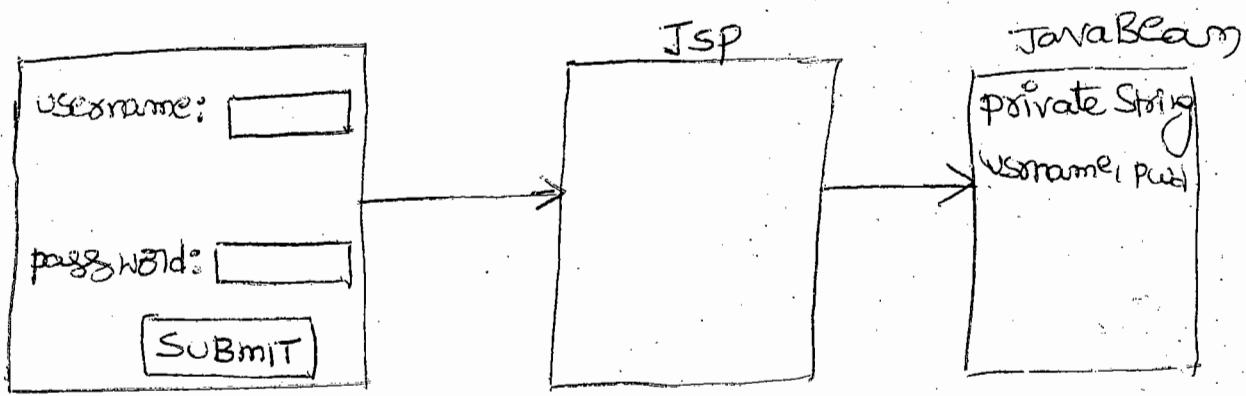
→ In order to create a JavaBean object in a JSP, we need the <jsp:useBean> tag!

→ To use a Java class as a JavaBean, the following rules are required:

- (i) The class must be stored in a package.
- (ii) Class must be public.
- (iii) Class should implement Serializable interface.
- (iv) A public default constructor.

(10) another debt negotiation

Date ex:)  
5/8/11.



`<jsp: SetProperty>` must be used inside  
`<jsp:useBean>`.

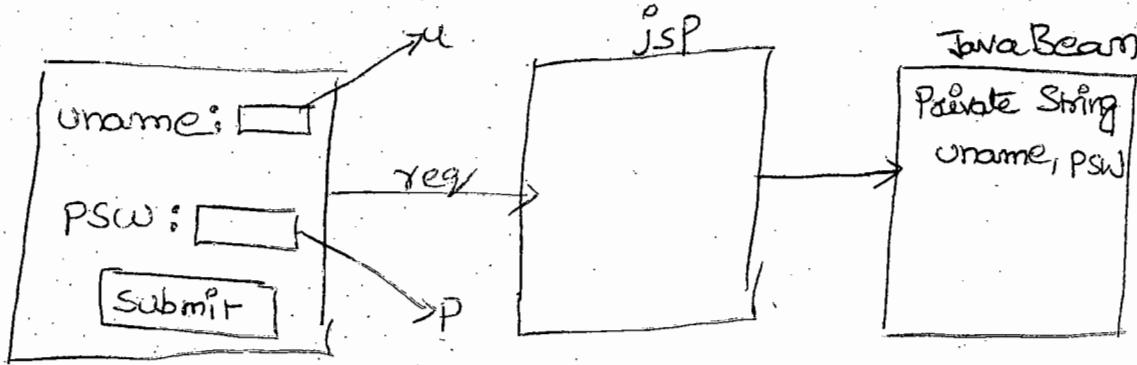
In the above example, request parameter names & Bean class variable names both are same. So we used `property="*"`

```
<jsp:useBean id="id1"
class="pack1.LoginBean">
```

```
<jsp:SetProperty name="id1"
property="*"/>
```

```
</jsp:useBean>
```

Cx-2



<?jsp:useBean id="id1" :

class = "pack1.LoginBean"

<?jsp:setProperty name = "id1" :

property = "uname"

param = "u" />

<?jsp:setProperty name = "id1" :

property = "pwd"

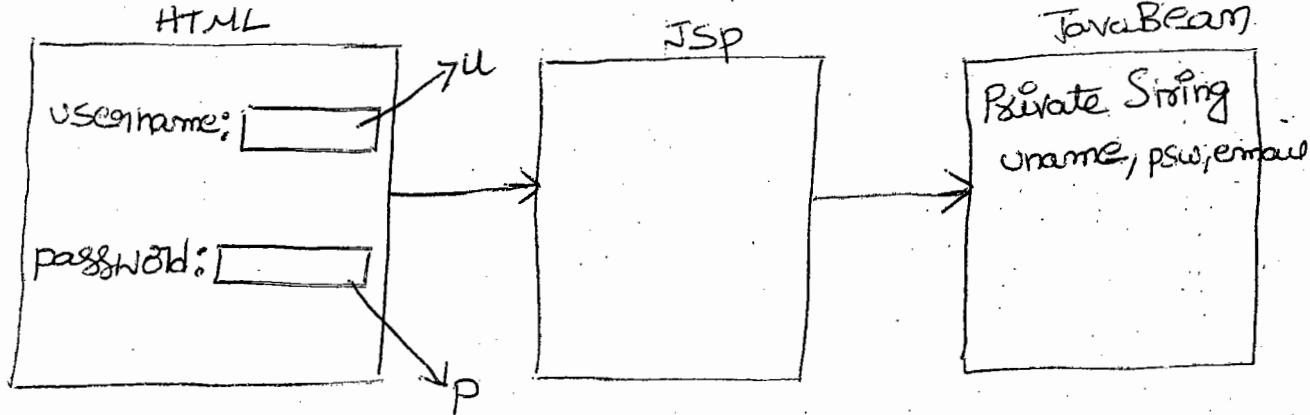
param = "p" />

</?jsp:useBean>

=> In the above example, request parameters names & Bean class property names can not match so we can not write property = \*,

so we have set properties individually an additional attribute called param is required

Ex: ③



```
<jsp:useBean id="id1"
class="pack1.LoginBean">
```

```
<jsp:setProperty name="id1"
property="uname"
param="u"/>
```

```
<jsp:setProperty name="id1"
property="pwd"
param="p"/>
```

```
<jsp:setProperty name="id1"
property="email"
value="abc@gmail.
com"/>
```

```
</jsp:useBean>
```

⇒ value attribute used to directly set the value into the JavaBean variable through

JSP.

`<isp: setProperty` totally contain 4 attributes they are

- (1) name
- (2) property
- (3) param
- (4) value

⇒ possible settings

- (1) name, property
- (2) name, property, param
- (3) name, property, value
- (4) name, property, param, value ✗ (not possible)

(3) `<isp: getProperty >`:

This standard action is used to reading value of a Java Bean property into a JSP whenever this tag is used in the JSP then getter method of the property is call.

→ If is not possible to read all properties of the javaBean at a time by calling the getter methods, it means we can not write `property="*"`.

- Always <jsp:getProperty> tag must be used  
at outside of <jsp:useBean>.
- ①
- only two attributes are allowed into <jsp:  
getProperty> those are name & property.

<jsp:getProperty  
name = "id" />

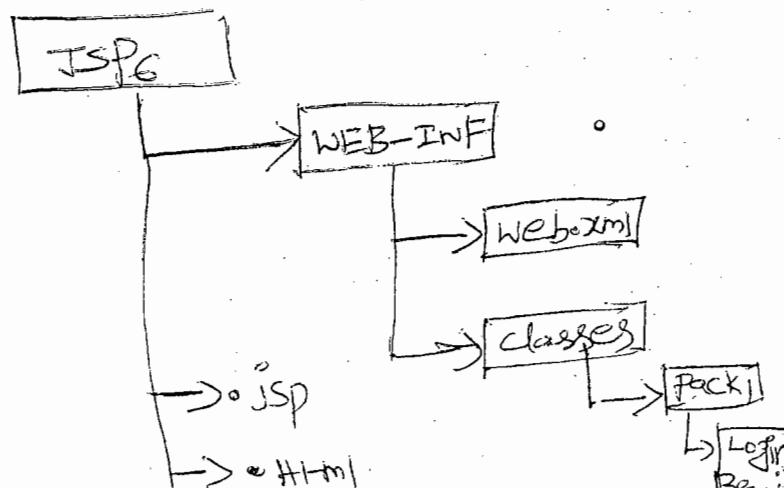
property = "name" />

<jsp:getProperty  
name = "id" />

property = "pwd" />

Ex:- The following example is from Jsp to  
JavaBean communication for setting values  
into the Bean and getting values from  
the Bean:-

Directory Structure



sed

## ① Login.html

<!-- login.html -->

<form action = "login.jsp">

username: <input type = text name = "uname">

<br>

password: <input type = password name = "pwd">

<br>

<input type = submit value = "click" >

</form>

## ② Login.jsp

<%-- Login.jsp --%>

<jsp:useBean id = "id1"

class = "pack1.LoginBean">

<jsp:setProperty name = "id1" property = "\*"/>

<jsp:useBean>

The username is: <jsp:getProperty

name = "id1" property = "uname"/>

<br>

\*... getProperty

## //LoginBean.java

package pack;

public class LoginBean

{

private String uname, pwd;

public void setUsername(String uname)

{

this.uname = uname;

}

public String getUsername()

{

return uname;

}

public void setPassword(String pwd)

{

this.pwd = pwd;

}

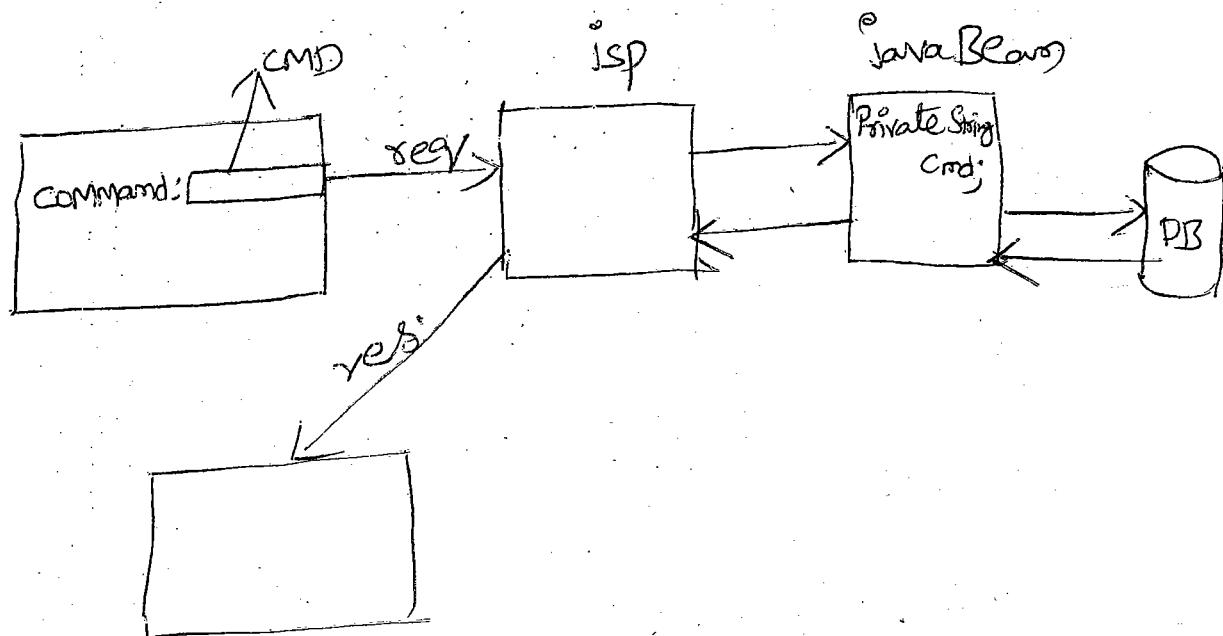
public String getPassword()

{

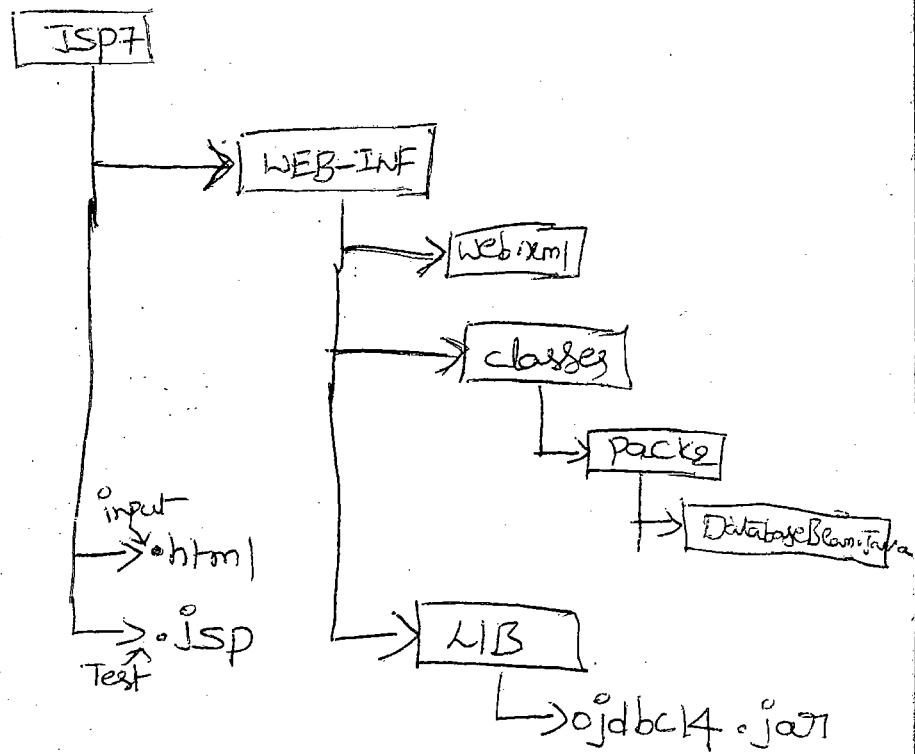
return pwd;

}

Exo 2



### directory structure



① <!-- content.html -->

<center>

<h1>

<form action="test.jsp">

Command: <input type="text" name="cmd"

><b>

<input type="submit" value="execute">

</form>

</h1>

</center>.

② <% -- Test.jsp -- %>

<jsp:useBean id="id1" class="pack2.DatabaseBean">

<jsp: setProperty name="id1" property="\*"/>

</jsp:useBean>

<%

int k = id1.bm();

if (k == -1)

{

out.println(" DDL command executed");

}

if (k >= 0)

{

out.println(" DML Command executed");

}

if (k == -10)

{

ArrayList al = id1.getData();

Iterator it = al.iterator();

while (it.hasNext())

{

Object o = it.next();

out.println(o.toString() + "<br>");

}

/>

}

<%@ page import="java.util.\*" %>

→ In the above JSP we are calling  
business method of JavaBean.

following 3 value

-1 → DDL cmd is executed

+ve → DML cmd is executed

-10 → select cmd is executed.

If select cmd Executed then we are calling method calling `getDatas()` of the Bean class to get an arraylist from the bean class.

Because of `ArrayList & Iterator` we have imported `java.util` package using `<jsp:directive` directive.

## 1) DatabaseBean.java

Package pack;

import java.sql.\*;

import java.util.\*;

public class DatabaseBean

{

private String cmd;

```
ArrayList al = new ArrayList();
```

```
public void setCmd(String cmd){}
```

```
 this.cmd = cmd;
```

```
public String getCmd(){}
```

```
{ return cmd; }
```

```
public int bm(){}
```

```
{ int m = 0;
```

```
try{
```

```
{
```

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

```
DriverManager.getConnection("jdbc:odbc:oradsh",
 "scott", "tiger");
```

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

```
if(st.executeUpdate(cmd))
```

```
{
```

```
ResultSet rs = st.getResultSet();
```

```
{
 al.add(rs.getString(1) + " " + rs.getString(2));
}

return -10;
}

else
{
 m = st.executeUpdate();
}

catch(Exception e)
{
}

return m;
}

public ArrayList getData()
{
 return al;
}
}
```

⇒ In the above bean class, we have used typeset driver, so we donot require any jar file into lib folder.

⇒ In the above bean class, we have called execute() method, because Enduser can type either select or non-select cmd.

⇒ execute() verifies whether the given cmd is select or non-select and then returns true for select operation, return false for non-select operation.

⇒ execute() need 2 supporting methods

(i) getResultSet()

(ii) getUpdateCount()

getResultSet() → called for select operation,

getUpdateCount() → " " Non-select "

⇒ In the above bean class we have used an arraylist for storing the rows of the ResultSet object.

we are returning ArrayList() to the JSP

\*\* We can not Return Directly ResultSet

Object from JavaBean to JSP because ResultSet object is not a Serializable object.

⇒ we have taken an arraylist, & arraylist is a serialized object we can transfer it back to JSP.

\*\*  
Note:

- (i) In Java, all collection objects are Serializable objects.
- (ii) In Java, legacy (old) collections are synchronized, but collection framework classes are not synchronized.

## Direct communication from Jsp to a database

(i) If we want to directly communicate from Jsp to a database then we should open the database connection, we should execute the cmd, finally we should close the connection within Jsp page itself.

(ii) In a Jsp if we want to open & close the database connection, we can override life-cycle `init()`, & `destroy()` methods of Jsp.

(iii) If we want to override `init()` & `destroy()` of a Jsp then we need to use Declaration tag of Jsp. We need to use scriptlet for inserting logic.

In: we need to use scriptlet for inserting logic required for the cmd execution.

Q: If we want to open the connection at `init()` & if we want to close it in the `destroy()`

then we need to declare connection, Stmt object as a global variable.

<% !

Connection con;

Statement stmt;

P v JspInit()

{

// open Connection

}

P v JspDestroy()

{

// close connection

}

%>

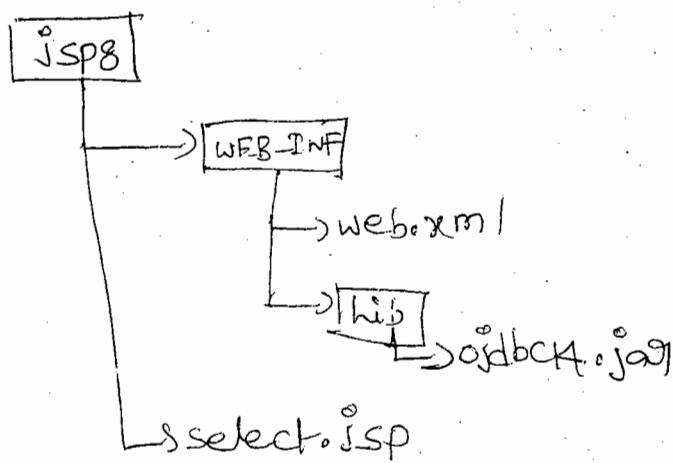
<%

=

logic.

%>

Directory Structure



<!-- select.jsp -->

<%@ page import="java.sql.\*" %>

<%!

Connection Con;

Statement Stmt;

public void JspInit()

{

try

{

Class.forName("oracle.jdbc.oracleDriver");

Con = DriverManager.getConnection("jdbc:

oracle:

Stmt = Con.createStatement();

}

catch(Exception e)

{

e.printStackTrace();

}

}

%>

jsp

<@>

```
try
{
```

ResultSet rs = stmt.executeQuery

( "select \* from dept" )

</>

<table border=2>

<@>

while (rs.next())

{

out.println (<tr>

out.println ("<td>" + rs.getString(1) + "</td>")  
||

(" " + " " || (2) + "</td>")

" " " " " " " " (3) " " )

out.println ("</tr>")

}

</>

</table>

< /o >

rs.close();

}

catch (Exception e)

{

e.printStackTrace();

}

</o >

< o !

public void jspDestroy()

{ my

{ con.close();

stmt.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

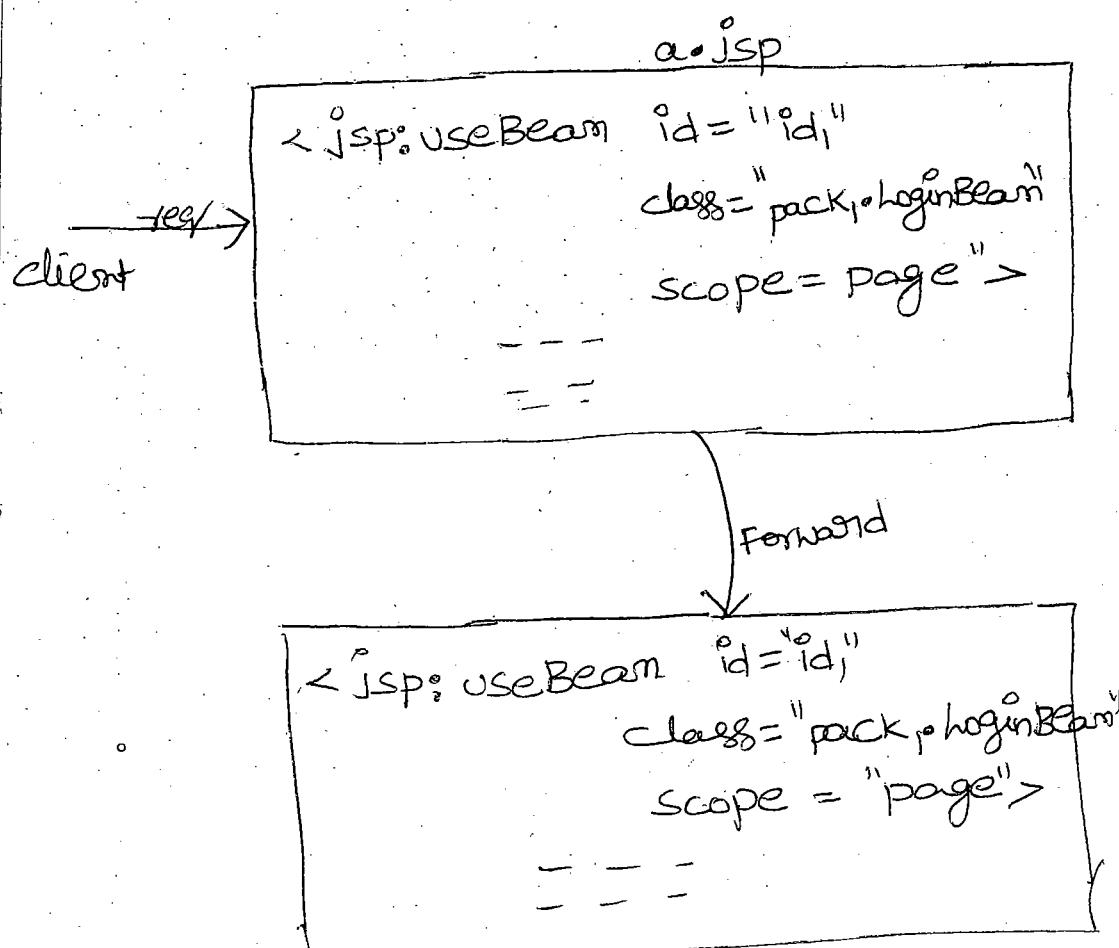
</o >

date

## = Bean Scopes

### page scope (Local or Default Scope)

- (i) page is the least scope of a bean in JSP.
- (ii) If a bean object is created in page scope that object is accessible only within that JSP.
- (iii) page scope object becomes local to the JSP.
- (iv) When a request is given to the JSP, the bean object is created and then control goes out of the flat JSP page either to the next page back to the browser that object will be removed from the server.
- (v) If we don't specify scope attribute then by default pagescope is assigned to the bean object.

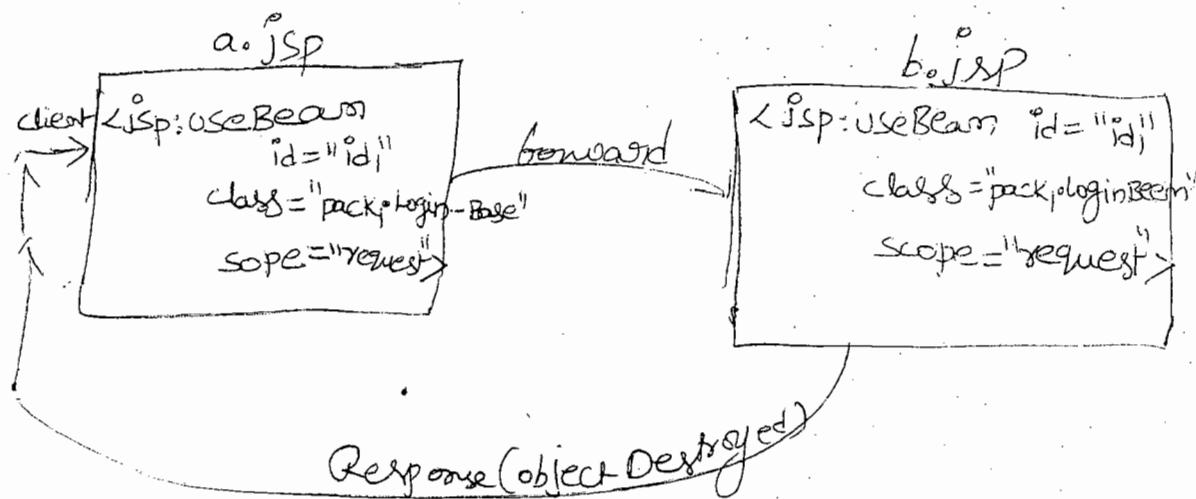


- In the above example client request the
- In the above case in `a.jsp`, the bean is given to `a.jsp` and it is forwarded to `b.jsp`.
- In the above case in `a.jsp`, the bean object "`id1`" is created with page scope and when the control is forwarded to `b.jsp` the "`id1`" object of `a.jsp` is removed and in `b.jsp` another object by the bean created with the name `id1`.
- In the above example "`id1`" object is crea-

The object created in a.jsp not accessible in b.jsp because of page scope.

## (2) Request Scope:-

- If a bean object is created with request scope then that object is available in the same jsp & if the request is forwarded to another page then in that page also the object is available.
- Request scope is just more than page scope.
- Request scope object is removed whenever response is given back to the client.



- In the above diagram both a.jsp & b.jsp the Bean scope added as request.

→ when a client request is given to a.jsp, id  
object created with request scope. request  
forwarded to b.jsp, same object id is transfe-  
red to b.jsp also.

→ Request scope means the object is available in  
the current page and also continued to forwarded  
page.

→ In the above diagram response is given  
back to the client the object "id" is removed  
from the server.

never → In both a.jsp & b.jsp same object "id" is  
used. Separate objects are not created.

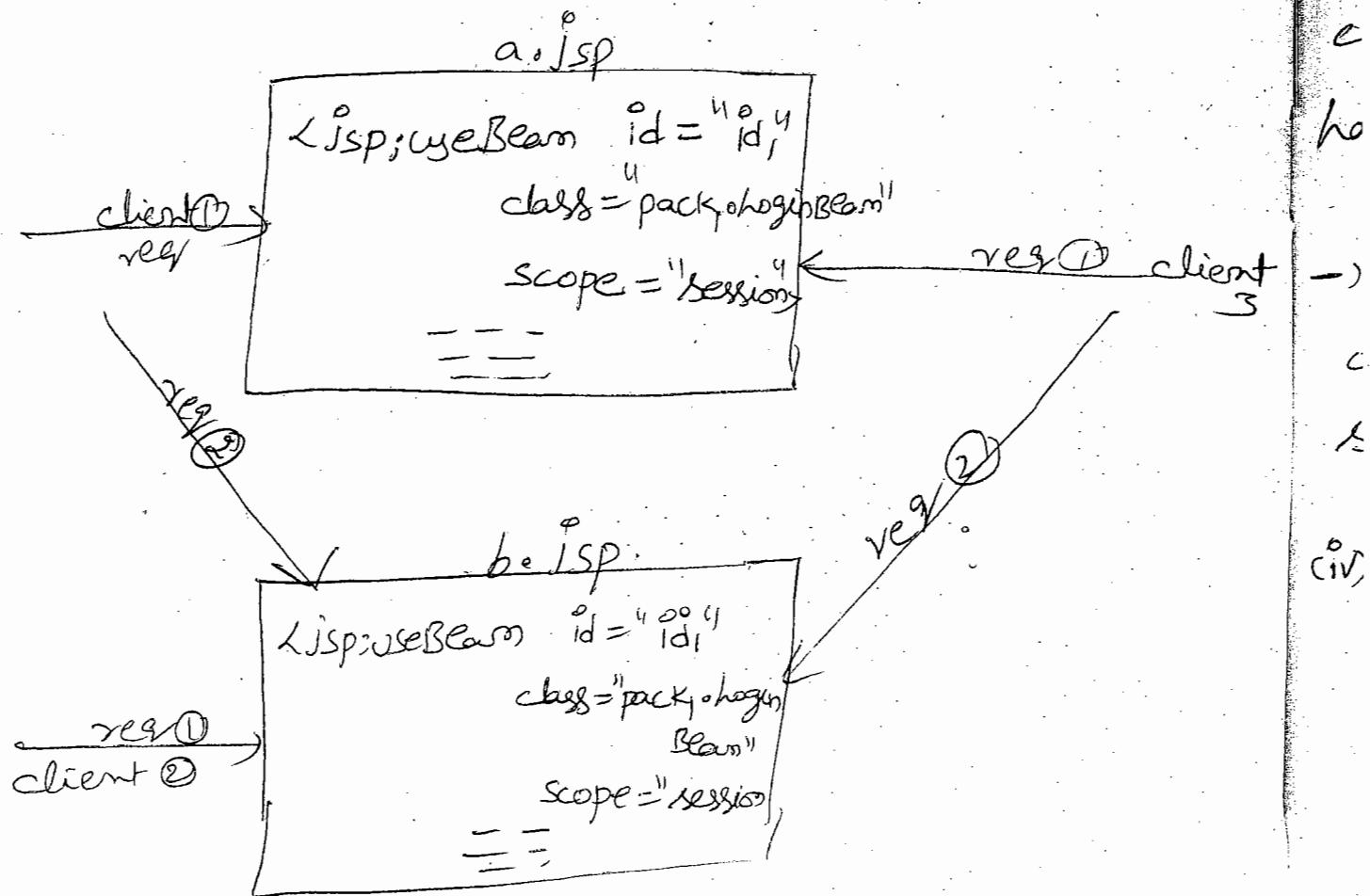
### (3) Session Scope

→ When a Bean object is created in Session  
scope then object will be created for each  
session separately.

→ For each new client, a new session  
starts & for each new session a new object  
of the bean class is created.

→ The Bean object is removed whenever

→ Session scope is just more than request scope.



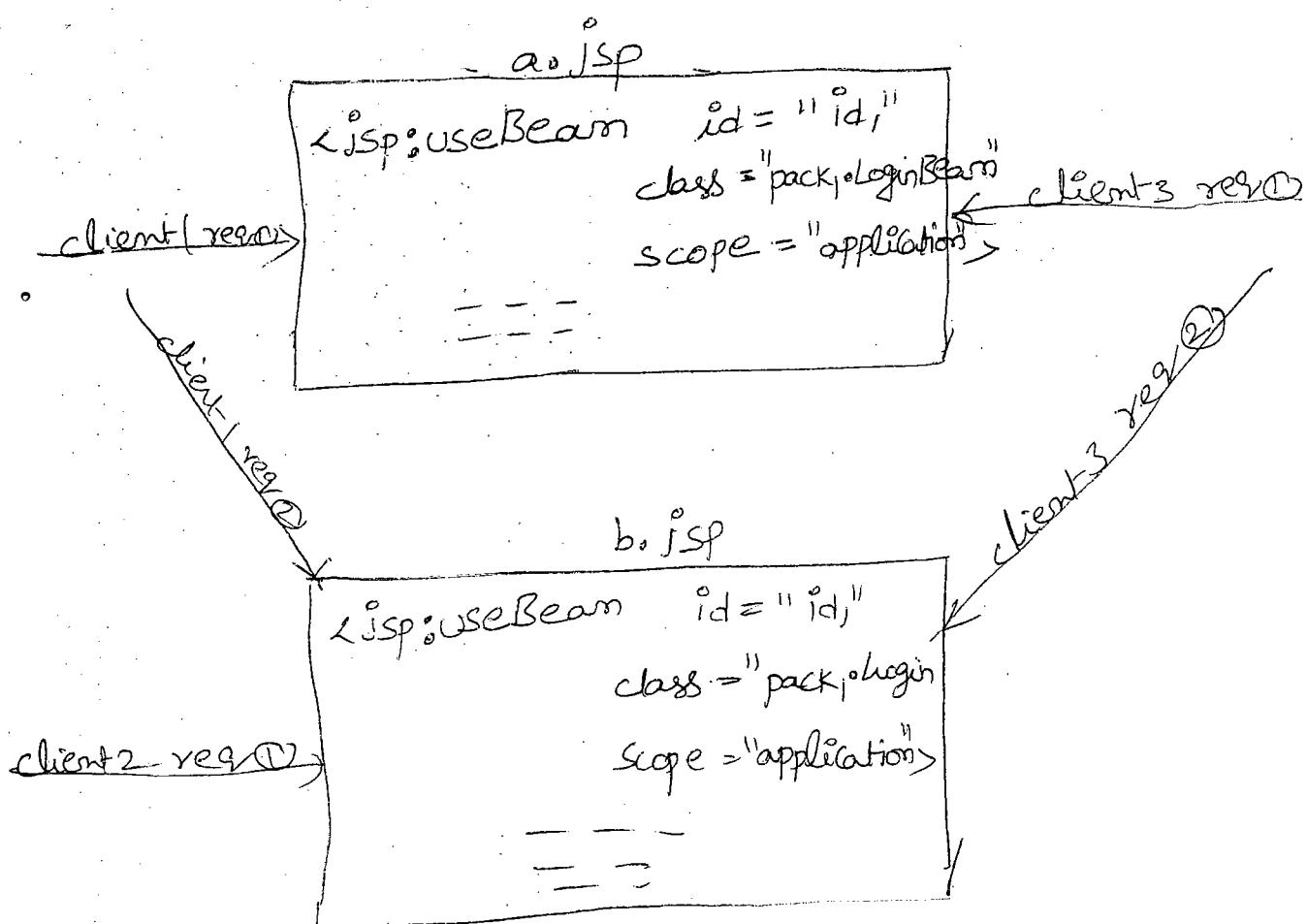
→ In the above diagram client ① & client 3 are sending first request to `a.jsp` & client 2 is sending first request to `b.jsp` for each new client a new session is started & the `id` object is created by each client (session) separately. In the above case `Id1` object is

est. Created for 3 times because of 3 clients (sessions)

→ For client 1 & client 3, in b.jsp the object "id;" is not created, because both are already existing clients to the "b.jsp", and the clients are already having their sessions with "id;" objects.

Client 3 → The life-time "id;" object is until that client session is expired, but not until response is given.

#### (iv) Application Scope:-



→ If a Bean object is created in this scope  
the object become Global to the application

→ In application scope for any number of clients & for any number of JSP's only one object is created and it is sharable across all the JSP's in the web-application

→ The highest scope of a bean object is application scope.

→ In the above diagram for all the clients & all the requests & all the pages only one object "id" is created it is available to all JSP's because of application scope.

\*\* what is the diff b/w page & application scope?

→ page scope is local & separate object is created for each page. application scope is global and it is one time created & sharable across all pages.

scope      \* what is the diff b/w page & request scope?

A:- In page scope if the request is forwarded to the next page, the object will be removed and it is not accepted in the next page.  
→ In the request scope the object is accessible to the next page if the request is forwarded to the next page.

\* what is the diff b/w session & application?

\* what is the diff b/w session & application?  
→ In session scope object is created separately for each session.

→ In application scope common object is created for all sessions.

\* what is the diff b/w request scope?

\* what is the diff b/w request scope?

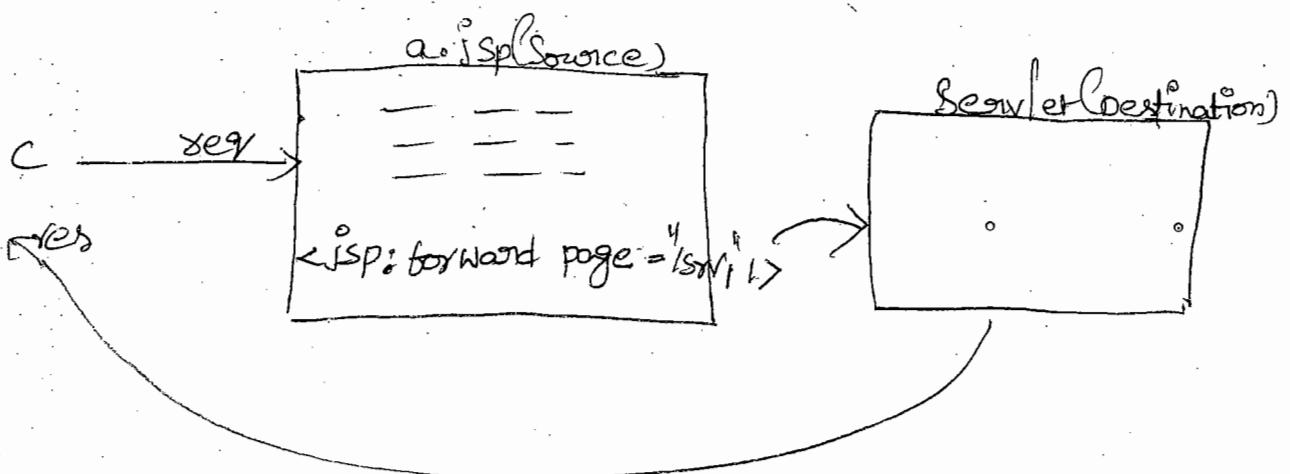
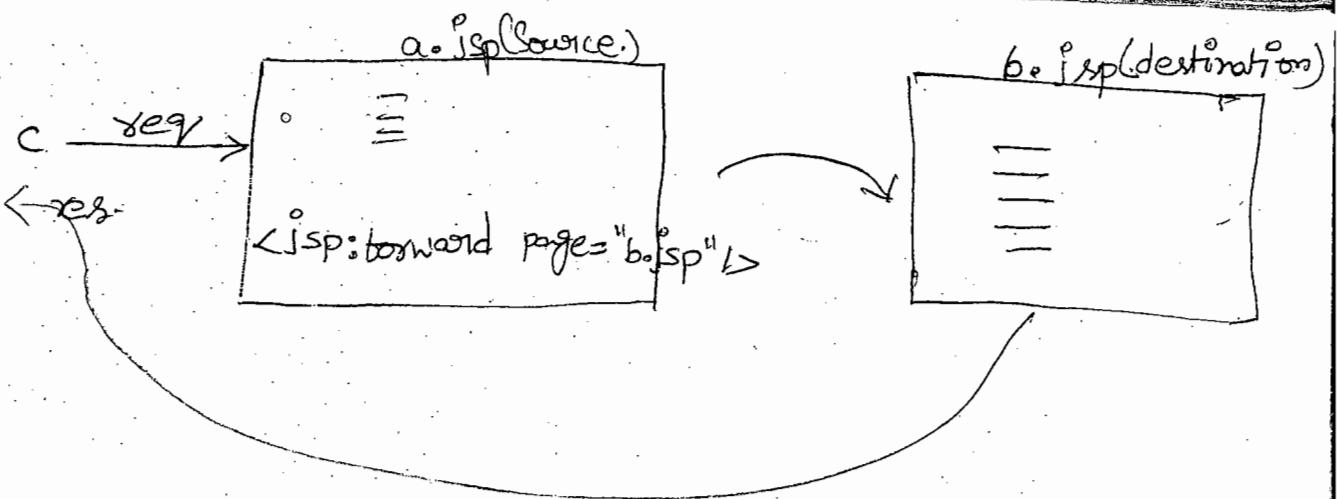
\* Request scope object life-time is completed until that request is completed.

In session scope object life-time is

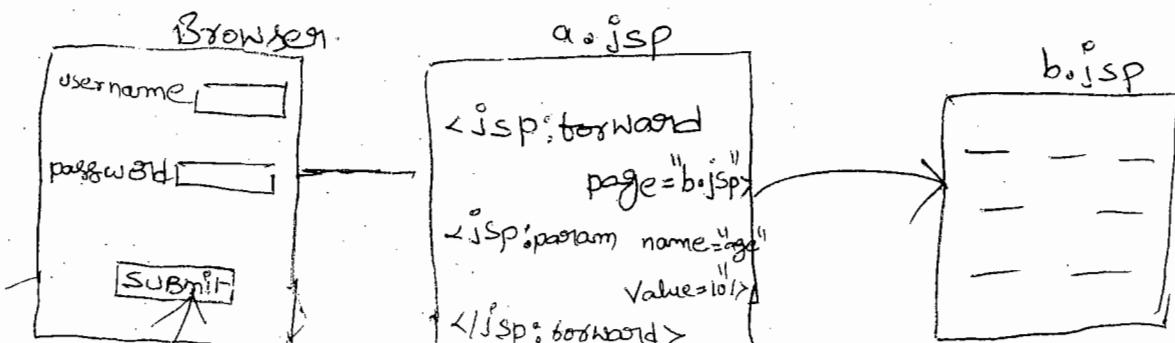
until that client session expired.

### <JSP:forward>

- (i) This standard action is used to forward a client request from one page to another page.
- (ii) This standard action internally uses request-dispatchers forwarding mechanism.
- (iii) While forwarding a client request, the Destination Resource can be a JSP page or a Servlet or an HTML page.
- (iv) While forwarding a client request, the source & destination resources must be within the same server.
- (v) While forwarding a client request, if destination is a JSP page then we should pass name of the JSP, if it is a Servlet we should pass the url-pattern of the Servlet



- within → while forwarding a client request, automatically the Request will be forwarded along with the Request parameters also forwarded.
- with → Along with the request parameters, we can also add additional parameters will be forwarded the request this can be done by using <jsp:param>



→ In the above diagram while forwarding the request from a.jsp to b.jsp automatically username & password are forwarded and along with that additional parameter age is also forwarded to b.jsp.

Sa

→ If we want to pass more than one additional parameters we should put more than one param tags inside the params.

```
<jsp:forward page = "b.jsp">
```

```
<jsp:params>
```

```
 <jsp:param name = "age" value = "10" />
```

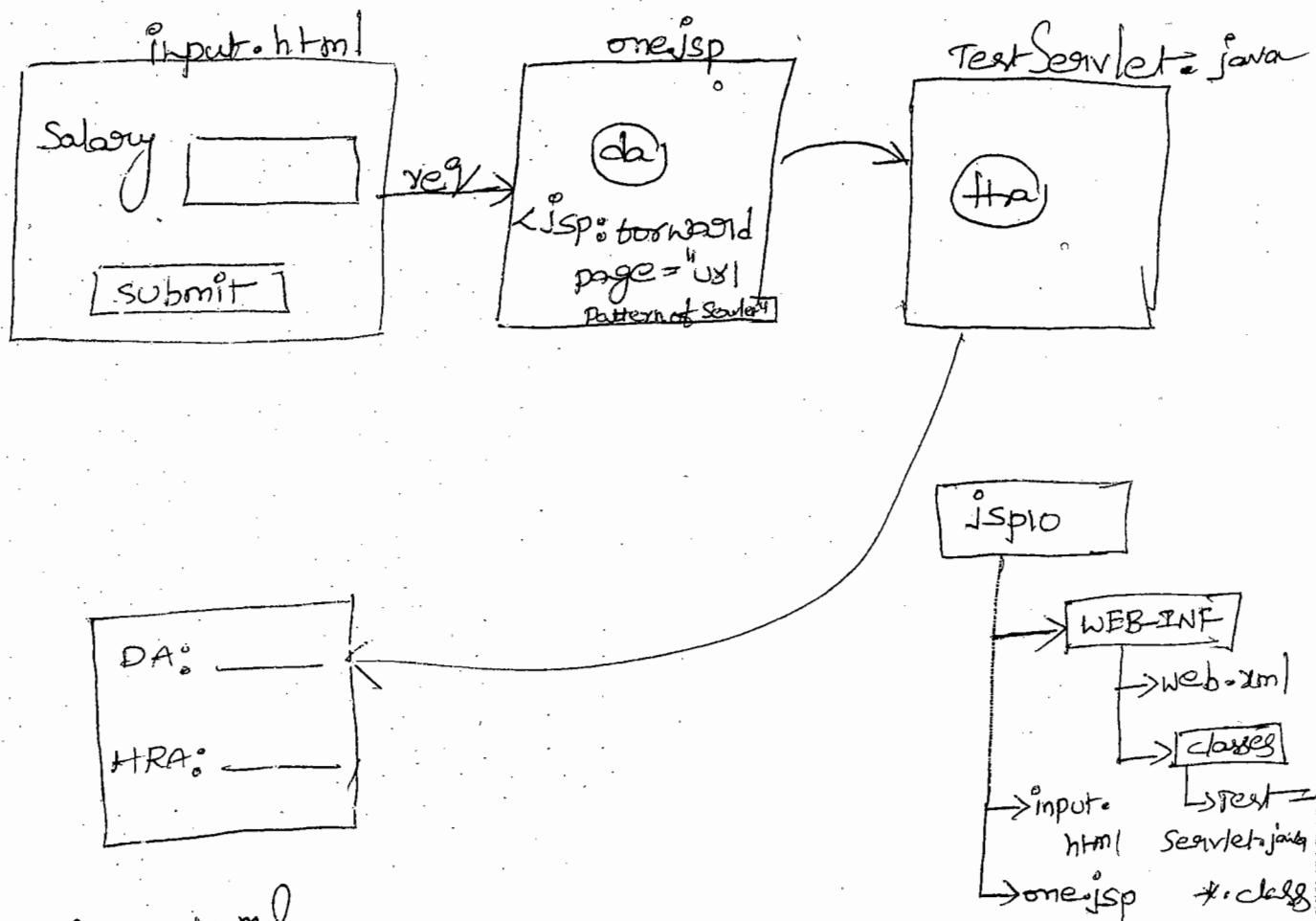
```
 <jsp:param name = "email" value = "-" />
```

```
<jsp:params>
```

```
</jsp:forward>
```

We can not use <jsp:param> & <jsp:params> individually. that means these tags must be used inside other class.

→ the following example is for forwarding a client request JSP to a servlet for calculating da & HRA for the given Salary.



input.html

! -- input.html -->

<CENTER>

<form action = "one.jsp">

Salary: <input type = text name = "s"><br>

<input type = submit value = "click">

</form>

</center>

② <!-- one.jsp -->

<% !

double da;

%>

<%

String s1 = request.getParameter("s");

double sal = Double.parseDouble(s1);

da = sal \* 0.10;

request.setAttribute("da", new Double(da));

%>

<jsp:forward page="TestServlet" />

### ③ // TestServlet.java

import javax.servlet.http.\*;

import java.io.\*;

import javax.Servlet.\*;

public class TestServlet extends HttpServlet  
public void doGet(HttpServletRequest req, HttpServletResponse res)  
throws SE, IOException

{ String s2 = request.getParameter("s")

// here s2 containing salary

double sal = Double.parseDouble(s2);

```
double hra = sal * 0.15;
```

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

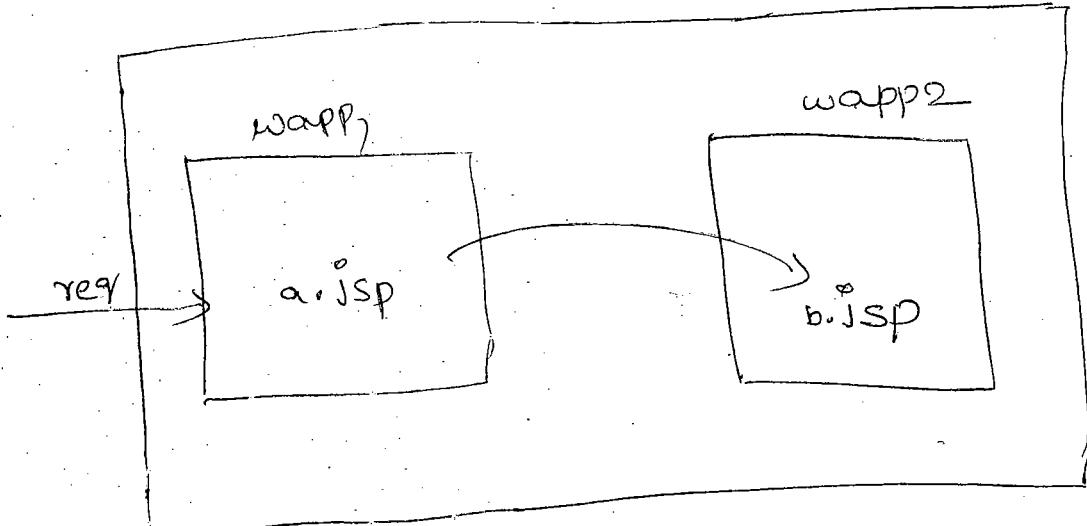
```
Double x = (Double) request.getAttribute("da");
```

```
x.doubleValue()
pw.println("DA": da);
```

```
pw.println("hra": hra);
```

```
}
```

while forwarding a request if destination resource  
is available in another web application; then  
we should mention web-application name also



```
<jsp:forward page="\wapp2\b.jsp"/>
```



Note: Marker interfaces are two types

(i) Pre-defined

(ii) user-defined | customized

\* tag we no that serialization can be done in two ways

(i) Complete or Total Serialization

(ii) Incomplete or partial "

\* whenever one class all variables are participated it is said to be complete persistency.

\* whenever some of the members are not participated it is partial persistency.

→ for they partial persistency we are using transient modifier then those members are not involved in serialization

Transient double avg.

→ Transient variables are non-static variable

Static members are not involved in the persistency.

