

# WWW.tutorial4us.com



College  
Projects

```
void main()  
{  
    int no;  
    clrscr();  
    printf("%d\n", no);  
}
```

Basic  
Program

C  
Tutorial



JAVA  
Programming



Tutorial4Us

Tutorials for Beginners



Tutorial4Us

Tutorials for Beginners



Tutorial4Us

Tutorials for Beginners

# tutorial4us.com

A Perfect Place for All **Tutorials** Resources

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

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

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

For More Tutorials Stuff Visit

# www.tutorial4us.com

A Perfect Place for All **Tutorials** Resources

# Design Patterns

(Natraj Notes)

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

Design pattern :-

Which comes as test solution for real time problems of application development.

Design patterns are best practices to use sw technologies effectively in project of application development.

→ Design pattern are Supporting code while developing sw projects by using plain technologies.

→ The worst solution for <sup>real</sup> ~~real~~ time problem of project or application development is called as "Anti-pattern".

→ The organization ISO maintains both design pattern and anti-pattern.

→ Design patterns are implementation takes place in coding phase of project development.

Generally PL/TL designs the task for developers specified the need and utilization of

Design patterns and developers will impliment them while doing Task completion.

→ There are 500+ design patterns in Java Environment.

→ Design patterns can be implemented by using many sw technology. (or) programming language

Jdk level design patterns :-

Singleton java class

Synchronized singleton java class

Factory method / Factory pattern

Abstract Factory

Template method

Builder pattern

prototype pattern

Flyweight pattern

IOC (Invasion of Control) pattern

Adaptes class

Fastline reader

~~PO class object / value class object~~  
V.O class / D.T.O class object

value object class / Date transfer object class.

Web level design patterns :-

View Helper

Composit view

MVC1 MVC3  
MVC2

Front Controller

Intercepting ~~Controller~~ <sup>Filter</sup>

Abstract Controller



## -: Integration Layer Design patterns :-

Session Facade  
Message Facade  
Service Locator  
Business Delegate

→ Senior developers designs solutions for reoccurring problems of technology utilization whereas the junior developers take best solutions out of them and Uses that as designed patterns.

## -: Model Layer :-

DAO (Data Access Obj).  
Abstract DAO

## ① Singleton Java class

Problem :- Instead of creating multiple objects of Java class having same data and existing ~~using~~ memory, degrading performance. It is recommended to create one object and use it ~~using~~ for multiple times.

Solution :- Use Singleton java class.

The Java class <sup>that</sup> allows this to create one object per JVM is called as Singleton java class.

→ The logger class of the Log4j API is given as Singleton java class.

→ The Service ~~locate~~ locator class will be implemented as Singleton java class.

## Rules to implement Singleton java class :-

- ① it must have only private constructors.
- ② must have <sup>private</sup> static reference variable of same class
- ③ override clone() to suppress cloning process.
- ④ must have <sup>public</sup> static factory method having the logic of Singleton. (create and return only one object).

All these rules close all the doors of creating objects for java class and opens only one door to create object (i.e. Factory method where Singleton logic is placed).

→ The method of a class i.e. capable of creating and returning same or other class objects is called as Factory method

// SingletonTest - Java

```
class STest
{
    // static reference variable
    private static STest n=null;
    // private constructor
    private STest()
    {
        S.o.p("STest: @-param constructor (private)");
    }
}
```

// Static Factory method

```
public static STest create()
{
    // object creation having Singleton logic
    if(n==null)
        n=new STest();
    return n;
} // class
```

public class SingletonTest

```
{
    P.S.V.M (String[] args)
    {
        STest t1=STest.create();
        t2 =
        t3 =
        S.o.p("t1 obj hashCode is " + t1.hashCode());
        S.o.p("t2 " " " + t2.hashCode());
        S.o.p("t3 " " " + t3.hashCode());
        STest t4 = (STest) t3.clone();
        S.o.p("t4 obj " " " + t4.hashCode());
    }
}
```

> javac SingletonTest.java

> java SingletonTest.



→ java.lang.Runtime class is a pre-defined JDK level Singleton java class.

### ② Synchronized Singleton java class :-

When multiple threads act on single object then there is a possibility of getting multithreading issues. This chance is also there while working with the methods on Singleton java class. To overcome this problem it is recommended to design Synchronized Singleton java class. That means the static factory method and other user defined methods of that class must be taken as synchronized methods.

### ③ Factory method / Factory pattern :-

problem:- Using "new" keyword we can't create object with flexibility and by applying restrictions

Solution:- Use Factory method / Factory pattern

The method of a Java class i.e. capable of constructing and returning its own class (or) other class object is called as "Factory method".

There are two types of factory method.

- 1) Static factory method
- 2) Instance factory method.

Examples on static factory methods :-

Thread t = Thread.currentThread();

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

Runtime rt = Runtime.getRuntime();

Calendar cl = Calendar.getInstance();

↳ it is not calendar class object. it is object of gregorian calendar class which is subclass of

Calendar class (abstract class).

→ static factory methods are useful to create object of java class outside of its class, that class contains only private constructors. These are also useful while designing Singleton java classes.

prototype of static factory method :-

```
public static <class name / abstract classname> interface name > method (param name)
```

↳ it is recommended

Examples on instance factory method :-

String s = new String("ok");

String s1 = s.concat("hello"); // okhello

```
StringBuffer sb = new StringBuffer("hello");
```

method of String class returns String class obj

```
String s2 = sb.substring(0,3); // hell
```

```
Date d = new Date();
```

```
String s3 = d.toString();
```

→ If u want to create 'new' object by using existing obj data and behaviour then use Instance factory method.

→ Factory method can return either its own class object (or) other class object (or) subclass object

→ The object given by factory method represents certain process / logic i.e. executed in factory method

→ better not to ~~design~~ <sup>design</sup> factory methods returning empty objects (objects containing default ~~value~~ <sup>data</sup>)

Application:-

For example code on Factory pattern / Factory method refer page no 1 & 2

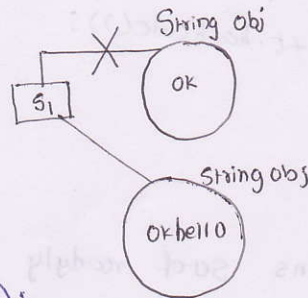
Prototype of Instance Factory method :-

```
public <classname> <interface name> / <abstract classname> method (-)
```

```
String s1 = new String("ok");
```

```
String s1 = s1.concat("hello"); // okhello
```

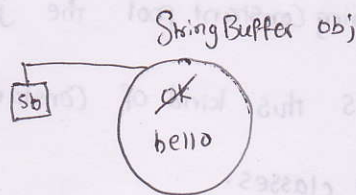
(String class is immutable class)



```
StringBuffer sb = new StringBuffer("ok");
```

```
sb.append("hello");
```

(StringBuffer class is mutable class)



most of the predefined class programmer developed java classes are generally mutable classes.



```

// Test.java
class Test
{
    int a;
    float b;

    test(int a, int b)
    {
        this.a = a;
        this.b = b;
    }

    public void modifyData()
    {
        a = a + a;
        b = b + b;
    }

    public void disp()
    {
        s.o.p("a=" + a + "b=" + b);
    }
}

```

// class

```
public class MutableTest
```

```
{
    psvm (String[] args)
    {
```

```
Test t = new Test(10, 20);
```

```
t.disp();
s.o.p("object hashcodes: " + t.hashCode());
t.modifyData();
```

```
t.disp();
s.o.p("object hashcode: " + t.hashCode());
} // main
```

```
} // class
```

NOTE :- When object data is modified if it is reflecting with the same object. Then it is called "mutable object" (its class is called mutable class).

→ ~~if~~ object data is modified if it is not reflecting with the current object but reflecting in a newly created object and reflecting by returning new object, then that object is called as immutable object (Its class is called as immutable class).

④ String Constant Pool :-

→ String constant pool maintains set of readily available objects. Due to this objects availability new objects will not be created they will be accessed from pool as needed.

→ To utilize this String constant pool the java.lang.String class is given as immutable class for remaining classes this kind of constant pools are not available, so, they are given as mutable classes.

→ To develop user-defined immutable java class :-

→ class must be final class.

→ member variables must be taken as private, final variable

→ when data modification is required it should be done through factory methods having logic to return new objects.

01/09/2012



//immutable Test.java

```
final class Test
{
    private final int a;
    private final String b;
```

//constructors

```
public Test(int a, String b)
{
    this.a = a;
    this.b = b;
    S.O.P ("Test; 2-param Constructor");
}
```

//write factory method when Data modification is required.

```
public Test modifyData (int a, String b)
{
    return new Test(a,b);
}
```

```
public Test modifyA (int a)
{
    return new Test(a, this.b);
}
```

```
public Test modifyB (String b)
{
    return new Test (this.a, b);
}
```

```
//display object data
public void disp()
{
    S.O.P ("a=" + a + "b=" + b);
}
```

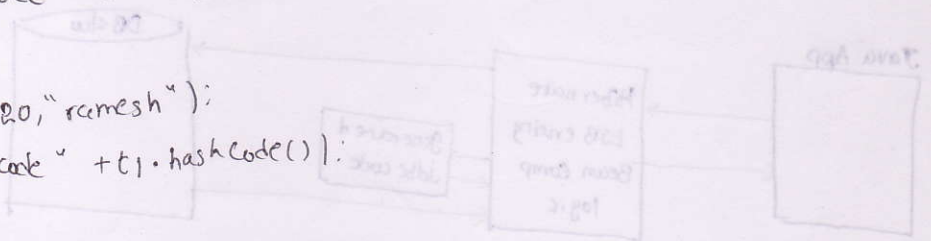
//class

```
public class ImmutableTest
{
    P S V m (String[] args)
    {
        Test t = new Test (10, "raj a");
        S.O.P ("t obj hashCode " + t.hashCode());
        t.disp();
        Test t1 = t.modifyData (20, "ramesh");
        S.O.P ("t1 obj hashCode " + t1.hashCode());
        t1.disp();
        t.disp();
        Test t2 = t.modifyA (30);
        S.O.P ("t2 obj hashCode " + t2.hashCode());
        t2.disp();
        t.disp();
    }
}
```

NOTE:- only final classes are not Immutable class.

Eg:- System, StringBuffer, Integer classes are final classes but they are not immutable classes.

//> javac ImmutableTest.java  
//> java Immutable



3 //main  
3 //class

\*) Template method Design pattern :-

problem :-

```
task 1 → a();
          b();
          c();
          d();
          x();
```

here to complete the task we need to multiple methods by remembering method names and their sequence of invocation.

Solution :-

use template method

```
public void mymethod()
{
    a();
    b();
    c();
    d();
    x();
}
```

```
task 1 : mymethod();
```

now only one method needs to be call<sup>to</sup> complete task.

The process method of org.apache.struts.action.RequestProcessor class calls 19 process Xxx() methods in a sequence to complete the task.

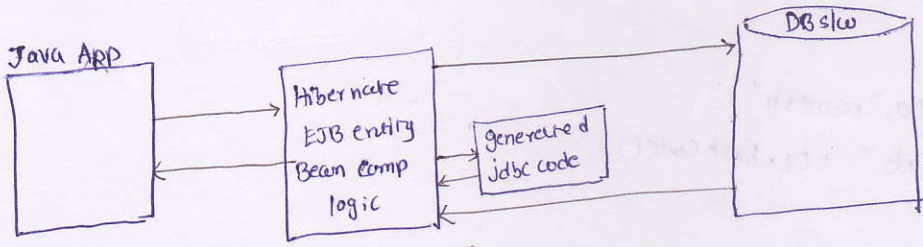
so, this method is called Template method.

→ ActionServlet should call all this 19 methods directly (or) RequestProcess class object to complete Request ~~Process~~. But it calls only one method i.e. process()

\*) FastLine Reader :-

problem :-

using high and ORM stubs (like hibernate) to develop persistence logic in our small and medium scale apps may degrade the performance because these ORM class internally generate jdbc code to complete the persistence logic development



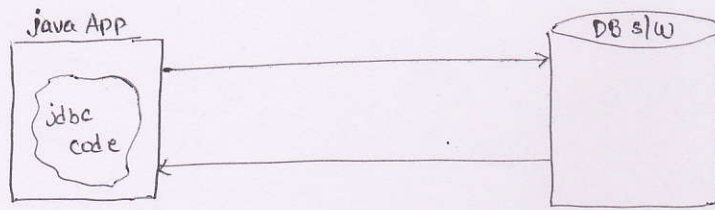
may degrade the performance.

Solution :-

Use FastLine Reader design pattern



Write jdbc code directly in a small and medium scale applications to interact with the db slow to improve the performance.



03/09/2012

Abstract :- hiding the implementation

⑦ Abstract Factory Pattern :-

problem :- while working with the object create and returned by factory method object factory pattern we need to specify the class names of those objects.

Solution :- use abstract factory pattern which is extension of factory pattern. Here, in the class names of objects returned by factory method / pattern to implement common interface (or) should extend from common class or abstract class.

abstraction means hiding implementation. abstract factory hides the class names of the object while receiving and utilizing those objects.

entire jdbc specification jdbc drivers development and utilizing those drivers in our jdbc code is happening based on abstract factory designed pattern.

```
Connection con = DriverManager.getConnection("jdbc:oracle:oci123", "scott", "tiger");
```

here con is jdbc con obj that it is the object of jdbc driver supplied java class that implements java.sql.Connection

here we are working with jdbc "con" obj without exposing and knowing its class name even though DriverManager.getConnection(---, ---, ---) returns different classes objects

based on jdbc driver and jdbc url we use. This is nothing but abstract factory pattern implementation.

Example: Example on abstract factory implementation.

// Common interface

```
interface ABC
{
    public void xyz();
}
```

// implement class 1

class Test1 implements ABC

```
{
    public Test1()
    {
        S.O.P("Test1");
    }
    public void xyz()
    {
        S.O.P("xyz of Test1");
    }
} //test1
```

// implementation class 2

class Test2 implements ABC

```
{
    public Test2()
    {
        S.O.P("Test2");
    }
    public void xyz()
    {
        S.O.P("xyz of Test2");
    }
} //test2
```

class Demo

```
{
    public public static ABC m1(String name)
```

```
{
```

// factory pattern

```
if (name.equals("a"))
    return new Test1();
```

```
else if (name.equals("b"))
```

```
    return new Test2();
```

```
else
```

```
    return null;
```

```
}
```





```

// client code
public class AbstractFactoryTest
{
    p s v m (String args[])
    {
        // abstract factory pattern behaviour
        ABC ab1 = Demo.m1 (args[0]);
        ab1.xyz();
    } // main
} // class

```

```

> javac AbstractFactoryTest.java
> java AbstractFactory a
> java AbstractFactory b

```

for related information and related example on abstract factory pattern refer pages no: 2

### Prototype :-

Problem:- Creating new object from scratch level having the existing object data has

initial data is ~~not~~ ~~actually~~ ~~required~~ quite time consuming process.

Solution:- Use cloning to create new object where new object contains existing data has initial data and constructor will not be executed during object creation.

⇒ When objects are created "new" keyword constructors will be executed for objects data initialization. ⇒ When objects are created to cloning constructors will not be executed. Because there is no need of objects initializing new objects that are created to cloning (as they old existing <sup>data</sup> data as initial data).

⇒ Working with cloning is nothing but working with prototype designed pattern.

```

// CloneableTest.java
class Demo implements java.lang.Cloneable
{
    int a, b;
    public Demo (int x, int y)
    {
        s.o.p ("Demo: 2-param constructor");
        a = x;
        b = y;
    }
}

```



```
public Demo()
```

```
{
```

```
    S.o.p("Demo: 0-Param constructor");
```

```
}
```

```
public Object myClone() throws Exception
```

```
{
```

```
    Object obj = super.clone(); // performs cloning current invoking object
```

```
    return obj;
```

```
}
```

```
} //class
```

```
public void disp()
```

```
{ S.o.p("a=" + a + " b=" + b);
```

```
}
```

```
public class CloneableTest
```

```
{
```

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

```
{
```

```
    // create first object
```

```
    Demo d1 = new Demo(10, 20);
```

```
    S.o.p("d1 hashCode: " + d1.hashCode());
```

```
    d1.disp();
```

```
    // perform cloning
```

```
    Demo d2 = new (Demo) d1.myClone();
```

```
    S.o.p("d2 hashCode: " + d2.hashCode());
```

```
    d2.disp();
```

```
} //main
```

```
} //class
```

```
// javac CloneableTest.java
```

```
// java CloneableTest
```

backup > MAPPS > APPS > Design pattern > prototype > java CloneableTest.java

→ compared to normal object creation the object creation through cloning takes less time

because there is no constructor execution.

→ when objects are created through cloning and Deserialization the constructors will not be executed

for related information on prototype Design pattern refer page no's 11 & 12

## 9) Builder Design Pattern:-

Problem:- Construction of complex objects directly by taking one complex class is not recommended process.

Solution:- Construct complex objects from multiple simple objects step-by-step as needed. This is called builder design pattern, which improves the reusability of multiple individual objects by constructing the complex objects.

Example:-

// BuilderTest.java

```
class Burger
{
    public int price()
    {
        return 25;
    }
}
```

```
class Fries
{
    public int price()
    {
        return 15;
    }
}
```

```
class Drink
{
    public int price()
    {
        return 30;
    }
}
```

```
class MealBuilder
{
    public int calcPrice()
    {
        // builder pattern logic
```

```
return new Burger().price() + new Fries().price() + new Drink().price();
    }
}
```

```
class SnackBuilder
{
    public int calcPrice()
    {
        // builder pattern logic
```

```
return new Fries().price() + new Drink().price();
    }
}
```



```

public class BuilderTest
{
    p s v m (String[] args)
    {
        //complex object 1
    }
}

```

```

MealBuilder mb = new MealBuilder();
int val = mb.calcPrice();
s.o.p("the meal price is "+val);

```

```

//complex object 2
SnackBuilder sb = new SnackBuilder();
int val1 = sb.calcPrice();
s.o.p("the snack price is "+val1);

```

}  
}

```

1> javac BuilderTest.java
1> java BuilderTest

```

backup > NAPS > Apps > DesignPattern > BuilderPattern

→ The ActionMapping object of Struts1.x environment is created based on Builder Design pattern.  
 → The request, response objects of Servlet Programming will be created by container based on Builder Design Pattern.

For related information on Builder Design pattern refer page no: 8 to 11

Adaptes class :-

problem :- when class of the application directly implements interface it as provide implementation for all the ~~imple~~ methods of that interface even though ~~it is~~ not interested to provide implemen <sup>tabior</sup> for certain methods

```

interface XYZ
{
    public void a();
    public void b();
    public void c();
}
class Test implements XYZ
{
    public void a()
    {
        //
    }
    public void b()
    {
        //
    }
    public void c() // null method
    {
        //
    }
}

```

'test' is not interested to provide implementation for method c() but it is forced to define it as null method definition.

Solution:- take adapter class implementing interface and provide null method definitions for interface methods. so the class of Application can override its own choice method by extending from Adapter class.

```
class MyAdapter implements XYZ
{
    public void a() {}
    public void b() {}
    public void c() {}
}
class Test extends MyAdapter
{
    public void a()
    {
        //
    }
    public void b()
    {
        //
    }
}
class Test1 extends MyAdapter
{
    public void c()
    {
        //
    }
}
```

→ generally the adapter classes will be taken as abstract classes. Bcz these classes doesn't contain serious logics (contains null method definitions). This helps provide direct instantiation for Adapter classes.

→ java.awt.event.WindowAdapter is an adapter class for 3 event listener interfaces like WindowListener, WindowStateListener, WindowFocusListener

→ javax.servlet.GenericServlet class is partially an adapter class for javax.servlet.Servlet interface

Q:- where did you use interfaces and Abstract classes in ur projects?

Ans:- PL designs API specification of the project having rules and guidelines to programed. this process we uses abstract classes supplies both rules and guidelines and we uses interfaces to supply only rules (method declaration).

Sun micro system gives API specification (like jdbc, servlet & etc...) to Vendor Companies having



rules and guidelines to develop s/w's (like JDBC drivers and connectors). In this process abstract classes will be used. Supply both rules and guidelines and Interfaces will be used. Supply just rules.

- while developing adapter classes abstract classes will be taken. while developing business components Service Interfaces are Java Interface.
- provide Special Run-time capabilities to object to JRE marker interfaces are required.
- May ordinary Java class as Special component must be extended from Special classes, abstract classes are its must implements Special interface.

06/09/2012

10) VO  
class | DTO class :-

Problem:- ResultSet object is not serializable object. so, we can't send it directly over the network.

Solution:-

Approach 1:- Use Rowsets <sup>instead</sup> ~~instead~~ of ResultSet

(All Rowsets are serializable objects by default)

Approach 2:- Copy data of ResultSet obj to collection framework data structure and send that DataStructure over the network. (All collection framework DataStructures are serializable objects by default).

NOTE:- Since most of JDBC drivers are not supporting Rowsets so use Approach 2 to solve the above problem.

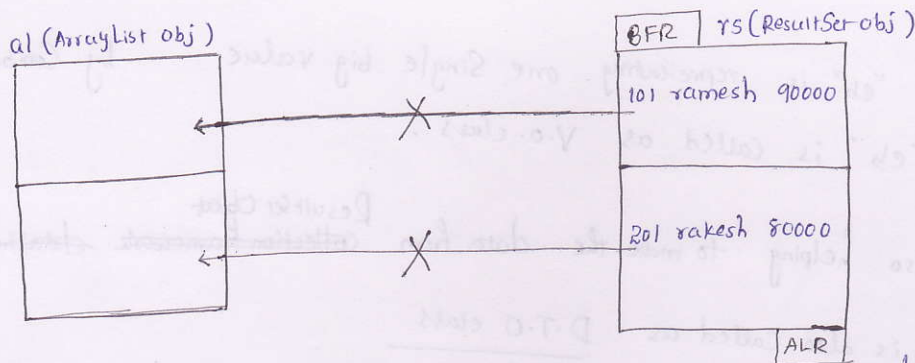
→ After moving data to data structure (ds) if you are looking to perform Simultaneous Read operations then use non-synchronize data structure like ArrayList, HashMap for better performance.

→ After moving data to collection framework data structure if you are looking to perform both read and write operations Simultaneously then use Synchronized data structure like Vector for thread safety.

in the above scenario we prefer working with ArrayList DataStructure.



understanding the problem related to copying ResultSet data to ArrayList



→ Each record of ResultSet contains multiple values including multiple objects. Each element of ArrayList allows only one object. So, we can't move each record of ResultSet directly to each element of ArrayList.

To make the above operation possible create multiple objects for User-defined class having the data of multiple records and add these objects to multiple elements of ArrayList. In this process this User-defined Java class is called as VO class / DTO class (Value Object) / (Data Transfer Object)

Example Scenario :-

VO class / D.T.O class (java Bean)

EmpBean.java :-

```
public class EmpBean implements java.io.Serializable
{
    private int no;
    private String name;
    private float sal;
```

// write setXxx(-) and getXxx() methods

-----  
-----  
-----  
-----

// Logic to copy ResultSet object records to ArrayList

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

```
ArrayList al = new ArrayList();
```

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

```
{
```

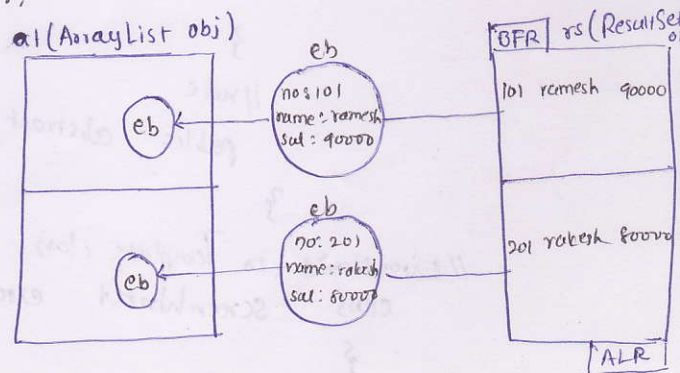
```
// copy each record to one EmpBean obj
```

```
EmpBean eb = new EmpBean();
```

```
eb.setNo(rs.getInt(1));
```

```
eb.setName(rs.getString(2));
```

```
eb.setSal(rs.getFloat(3));
```



//add each EmpBean object to ArrayList

al.add(eb);

} //while.

→ In the above scenario "eb" is representing one single big value by combining multiple values. So, the class of "eb" is called as V.O. class.

Similarly 'eb' is also helping to move the data from ~~Collection framework~~ <sup>ResultSet Object</sup> ~~datastructure~~ to ArrayList elements. So, 'eb' is also called as D.T.O class <sup>the class of</sup>

NOTE:- we need to implement above design pattern only by working with JDBC.  
This implementation is not required while working with Spring, hibernate think of they directly return List Data structure.

(ii) Design Pattern  
\*) Template class:-

problem:- Developing multiple classes of same category from scratch level kills the reusability and increases the burden on the programmers.

Solution:- Use Template class providing mould for programmers having rules and guidelines. So, programmers can develop their classes Based on this template class quite easily.

Template classes are generally abstract classes.

Example:-

```
//TemplateTest.java
```

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

```
//Template class
```

```
abstract class Writer1
```

```
{
```

```
    // guideline
```

```
    public int sum(int a, int b)
```

```
    {
```

```
        return a+b;
```

```
    }
```

```
    // rule
```

```
    public abstract void showResult(int res);
```

```
}
```

```
// Exception 1 to Template class
```

```
class ScreenWriter extends Writer1
```

```
{
```



```
public void showResult(int res)
```

```
{
```

```
    s.o.p(res);
```

```
}
```

```
} //ScreenWriter.
```

```
//Extension2 to Template class
```

```
class FileWriter1 extends Writer1
```

```
{
```

```
    public void showResult(int res)
```

```
    {
```

```
        try
```

```
        {
```

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

```
BufferWriter  
BufferedWriter
```

```
        bw = new
```

```
        BufferWriter (new OutputStreamWriter(new FileOutputStream("abc.txt")));
```

```
        bw.write(s1);
```

```
        bw.flush();
```

```
        bw.close();
```

```
    } //try
```

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

```
} //catch
```

```
} //class
```

```
public class TemplateTest
```

```
{
```

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

```
    {
```

```
        ScreenWriter sc = new ScreenWriter();
```

```
        int res = sc.sum(10, 20);
```

```
        sc.showResult(res);
```

```
        FileWriter1 fw = new FileWriter1();
```

```
        res = fw.sum(10, 20);
```

```
        fw.showResult(res);
```

```
    } //main
```

```
} //class
```

(12) IOC / Dependency Injection :-

Problem:- Dependency Lookup:- Dependency lookup of resource is getting its dependent

values by searching for them explicitly is called as Dependency lookup. In Dependency lookup the Resource has to pull the value explicitly before utilizing them.

Eg(1):- student gets metrical only when he demands for it.

Eg(2):- The way DataSource object is gathered from Registry s/w through JNDI code.

solution:- Dependency Injection. / (IOC).

Solution:- dependency Injection type.

Dependency Injection:- If underlying container | Framework | server | Run-time environment dynamically

Assigns values to resource then it is calls "dependency Injection"

→ In Dependency Injection underlying server | container... pushes the values to ~~resource~~ resource. so, ~~the~~ the resource need ~~not~~ to spend additional time to gathered the values.

Eg(1):- The way JVM executes constructors automatically to initialize the object ~~when~~ when object is created. Eg(2):- The way ActionServlet writes formdata to formbean.

Eg(3):- The way Spring Container performs Setter, constructor and Interface Injections on Spring Bean

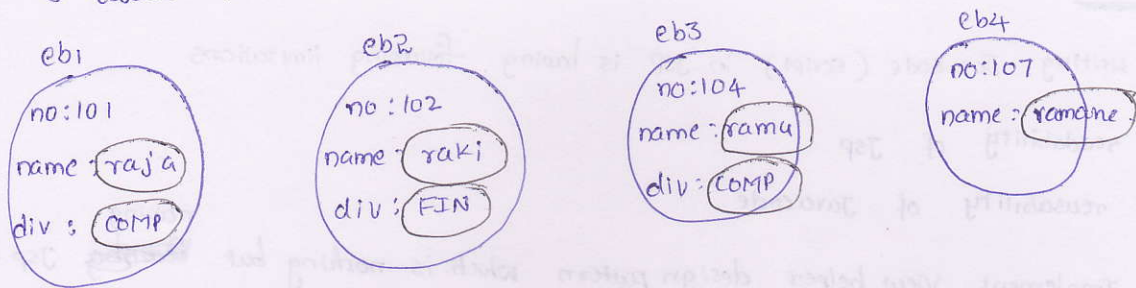
- The way object is initialize through constructor execution comes under Dependency Injection.
- The way object is assigned with data through method calls comes under Dependency lookup.

~~\*)~~ Fig weight

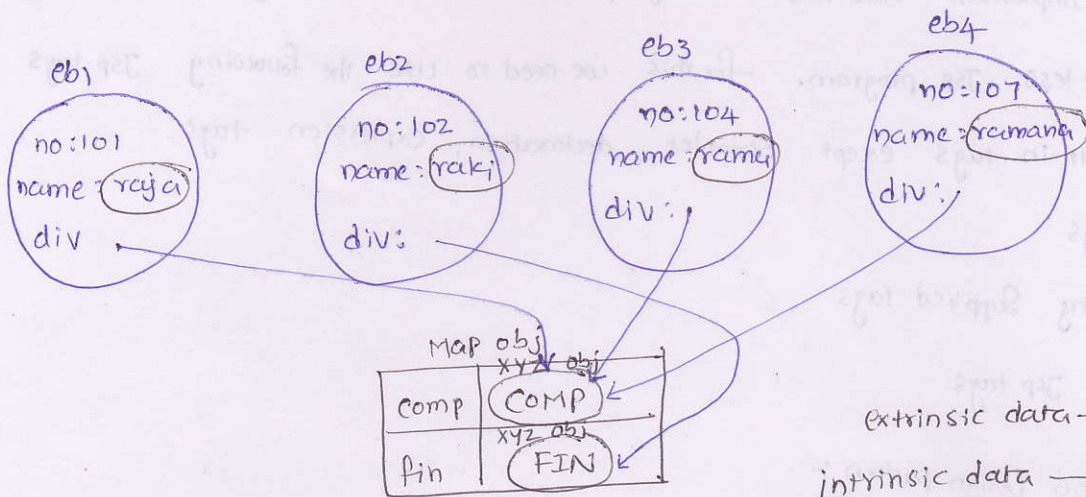


## Fly weight pattern :-

Problem:- 3 objects of employee class representing 4 employees details



Solution:-



Problem:- when you create multiple objects for a class having different values still

there is a possibility of having some same, common data/values in certain properties of these multiple objects. Instead object think about allocating common memory and using multiple objects.

→ The common data of multiple objects created for a class is called ~~intrinsic~~ <sup>intrinsic data</sup>

Similarly the data that is specific to each object is called extrinsic data.

In the above diagram no name values in every object comes under extrinsic data and "div" comes under intrinsic data.

Solution:- store extrinsic data in every object and intrinsic data in common memory (common obj) and use it in shared every object through Flyweight Design pattern implementation.

For example application on Flyweight Design Pattern, refer given supplementary handout.



## II) \* Web level Design pattern \*

### ① View Helper :-

Problem :- Writing Java code (script) in JSP is having following limitations

- ① kills the readability of JSP
- ② kills the reusability of Java code.

● Solution :- Implement view helpers design pattern which is ~~making~~ <sup>making</sup> JSP programs as

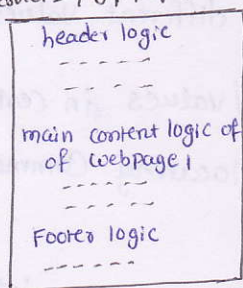
Java code less JSP program. For this we need to use the following JSP tags

- ① JSP built-in tags except Scriptlet, declaration, expression tags.
- ② JSTL tags
- ③ Third party Supplied tags
- ④ Custom JSP tags

### ② Composit View Design pattern :-

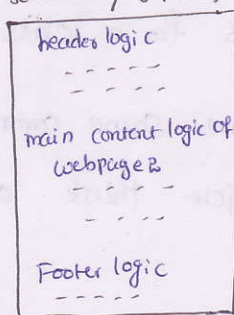
Problem :-

Servlet 1 / JSP prog 1



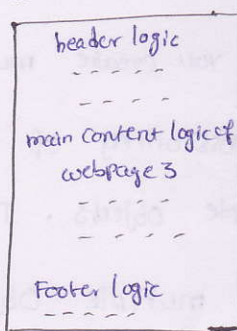
(generates webpage 1)

Servlet 2 / JSP prog 2



(generates webpage 2)

Servlet 3 / JSP prog 3



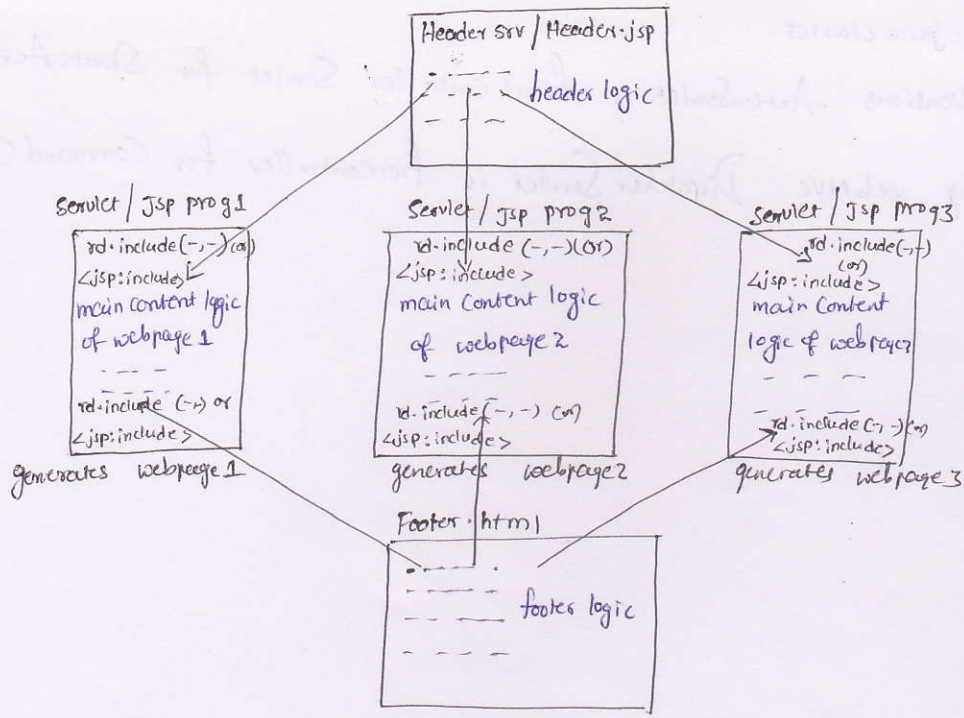
(generates webpage 3)

Here the common logic of multiple web resources program are not reusable logic (like header, footer) <sup>or</sup> because they are ~~not~~ hard coded in every web resource <sup>program</sup>.

→ The multiple web pages of web site contain same header and footer contents, But main content of <sup>each page</sup> webpage is different.

Solution :- make <sup>the</sup> common logics as reusable logics by keeping them in separate web resource programs and include their output. ~~to make~~ The main web resource programs its generate webpages for this we need to use `<include>` / `<jsp:include>`

this whole process is called implementation of Composite view design pattern.



NOTE:- here Common logics (header and footer logics are reusable logics)

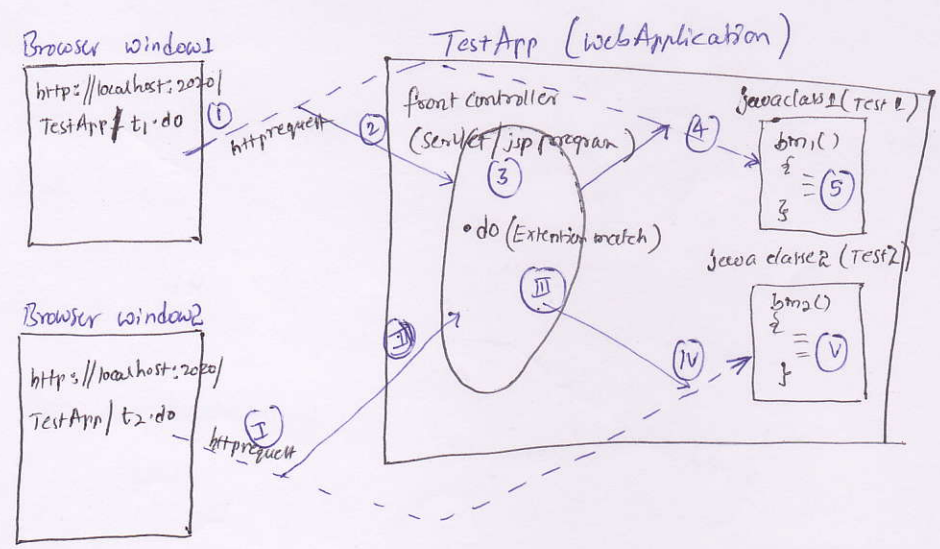
② Front Controller Design pattern :-

Problem:- Ordinary java classes that are acting as web resource programs of web applications can't take the client generated httprequest directly.

NOTE:- only Servlet/JSP programs of Java webapplication can take httprequest directly.

Solution:- Develop special Servlet/JSP programs as front controller trap all httprequest given by clients and to pass them to appropriate java classes

NOTE:- Front Controller Servlet/JSP program must be configured web.xml either by using extension match (or) directly match url-pattern.

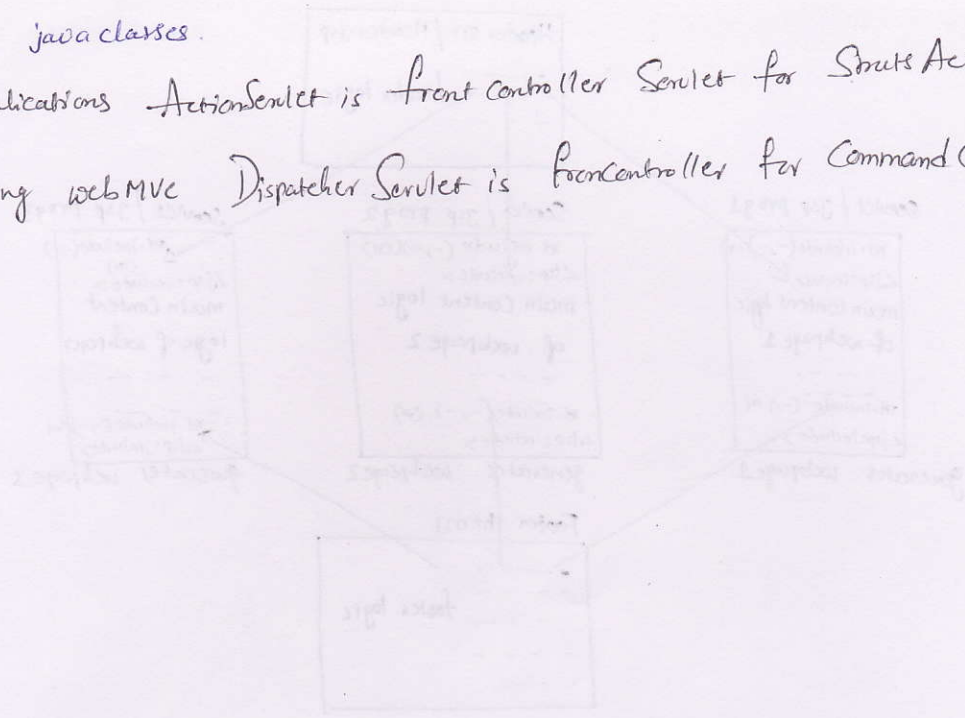


NOTE:- Front Control is a special web resource program (servlet program/ jsp program) which acts as Entry and Exit point for all request that are given to other webresource programs of



web application which are java classes.

Eg:- In Struts 1.x applications ActionServlet is front controller Servlet for Struts ActionClasses (Java classes). In Spring webMVC DispatcherServlet is front controller for Command Controller classes. (Java classes)



NOTE: front controller (front and back logic or separate logic)

front controller design pattern :-

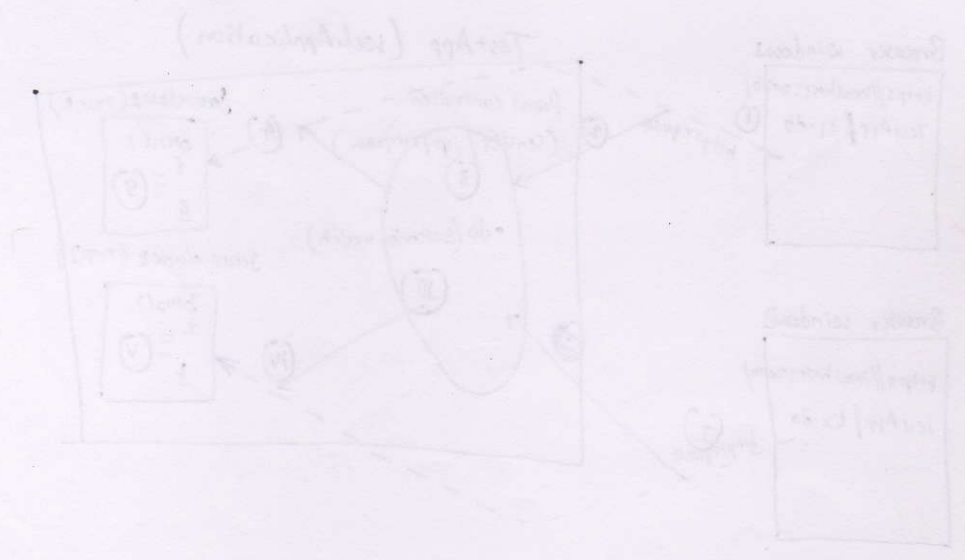
front controller - Central java class that act as a response program of web application and are the client generated request directly.

NOTE: only controller program of web application can be implemented directly.

front controller - Design front controller program as front controller that all request from client send to front controller to appropriate java class.

NOTE: front controller (controller) program must be configured in web.xml file of web application.

extension method (or) client request, url-pattern.



NOTE: front controller is a special response program (controller) that act as a front end for all request that are given to the response program of web application.



web.xml

Configure FrontSrv program in web.xml file with \*.do  
\*.do url pattern

NOTE:- Front Controller Servlet / JSP program must be configured in web.xml file either with directory match url pattern or with extension match url pattern.

web.xml

```
<web-app >
```

```
  <servlet >
```

```
    <servlet-name > F </servlet-name >
```

```
    <servlet-class > FrontSrv </servlet-class >
```

```
  </servlet >
```

```
  <servlet-mapping >
```

```
    <servlet-name > F </servlet-name >
```

```
    <url-pattern > *.do </url-pattern >
```

```
  </servlet-mapping >
```

```
</web-app >
```

// FrontSrv.java

```
import javax.servlet.*;
```

```
import javax.servlet.http.*;
```

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

```
public class FrontSrv extends HttpServlet
```

```
{
```

```
  public void doGet (HttpServletRequest req, HttpServletResponse res) throws
```

```
  {
```

```
    // generate settings
```

```
    PrintWriter pw = res.getWriter();
```

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

```
    // take the request and pass the request to appropriate java class
```

```
    int result = 0;
```

```
    if (req.getServletPath().equals ("/t1.do") )
```

```
    {
```

```

Test1 t1 = new Test1();
result = t1.bm1(10, 20);
} //if
else if (req.getHeader("t2.do") != null)
{
Test2 t2 = new Test2();
result = t2.bm2(10, 20);
}
// display the result
pw.println("<br><b>The result is : " + result);
} //doGet()
public void doPost (HttpServletRequest req, HttpServletResponse res)
{
doGet (req, res);
}
}
}

```

Date: 10/09/2012

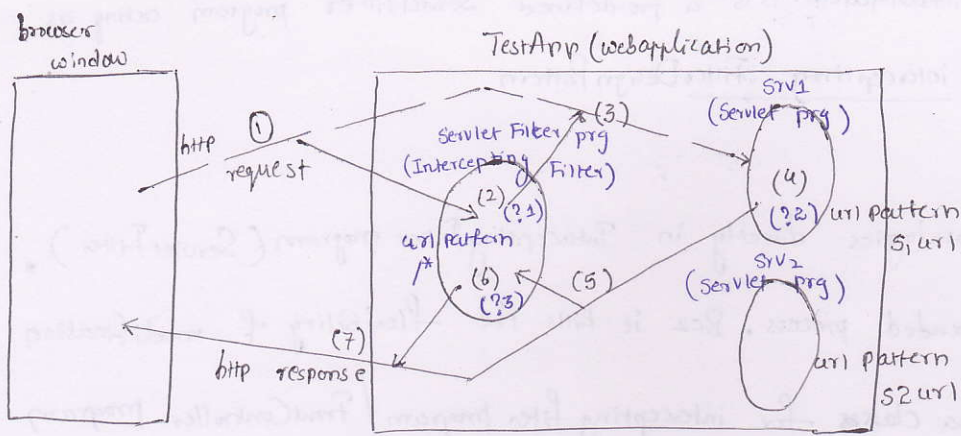
④ Intercepting Filter - Intercepting Filter the special web resource program of web application.

that is capable of trapping the request & response of other web resource programs. is called as "Intercepting Filter".

Problem :- keeping the common and global pre-request processing logic and post response generation logic in every main web resource program of web application kills the reusability of the code.

Solution :- Implement Intercepting filter designed pattern which is nothing but Servlet filter program. → Servlet filter program execute common pre-request processing logic by trapping request Similarly execute common post response generation logic by trapping the response of other web resource programs.





(?1) represents pre-request processing logic.  
 (?3) represents post-response generation logic.  
 (?2) represents the main-request processing logic.

→ Servlet Filter is nothing but implementing intercepting Filter Design pattern. Generally we place the following pre-request <sup>processing</sup> logic in Servlet Filter program.

- ① Authentication logic
- ② Authorization logic
- ③ logging logic

→ we place the following post-response generation logics ① Compression logic (It takes the output of compression the logic.)  
 ② Transformation logic and etc. ....

(NOTE 1) :- Front Controller Traps the request to pass them to ordinary java classes

Intercepting Filter Traps the request going to Servlet programs, Jsp programs and Java classes.

A Sample Servlet Filter program Development :-

MyFilter.java

```
public class MyFilter implements Filter
{
    public void init (FilterConfig fc)
    {
        ----- //initialization logic
    }

    public void doFilter (ServletRequest req, ServletResponse res, FilterChain fc)
    {
        ----- //pre-request processing logic
        fc.doFilter (req, res);
        ----- // post-response generation logic
    }

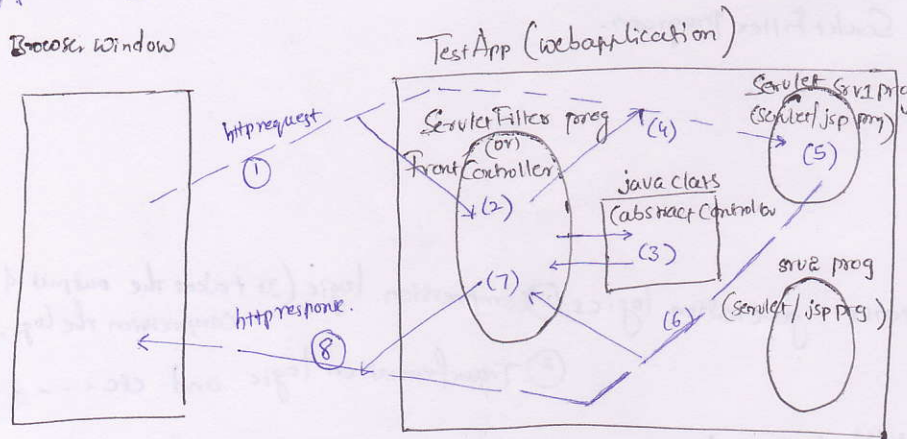
    public void destroy ()
    {
        ----- //uninitialization logic
    }
}
```

→ In Struts 2.x application FilterDispatcher is a pre-defined ServletFilter program acting as Controller. by implementing intercepting Filter Design Pattern.

⑤ Abstract Controller:-

→ Problem:- Hard Coding mainlogics directly in Intercepting Filter program (ServletFilter), FrontController is not recommended process. Bcz it kills the flexibility of modification

Solution:- Develop Helper Java classes for intercepting filter program / FrontController program having Mainlogics get the flexibility of modification. This Helper class is nothing but Abstract Controller.



→ Abstract Controller class is very useful to customize the logics of Front Controller program (3) Intercepting Filter program being from outside of these programs.

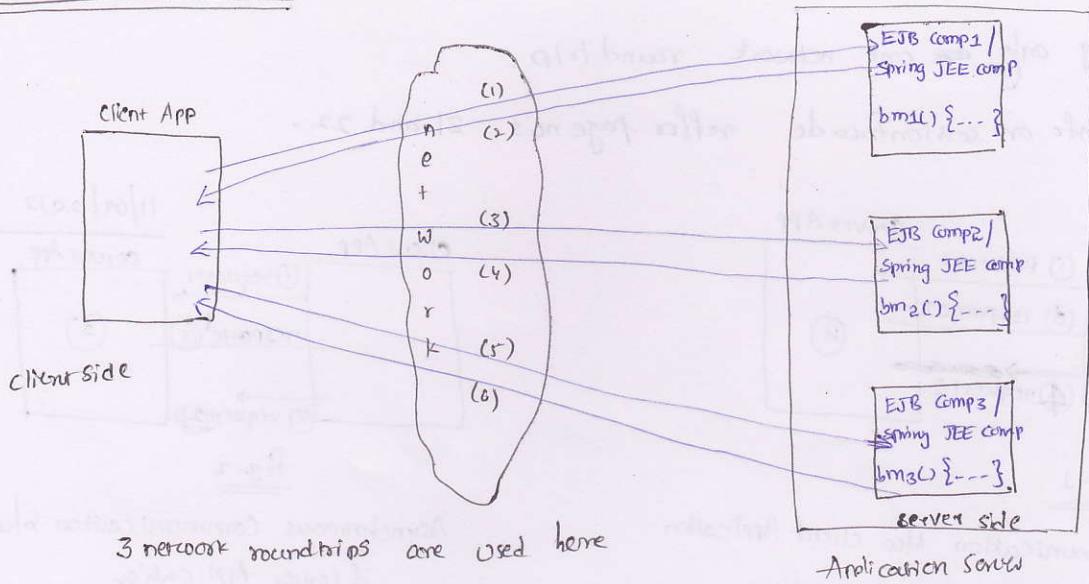
→ in Struts 1.x environment RequestProcessor class is Abstract Controller for the Front Controller ActionServlet. Similarly in Struts 2.x environment Interceptors are the abstract controllers for Intercepting Filter / Controller called FilterDispatcher.



III) Integration layers DesignPattern \*)

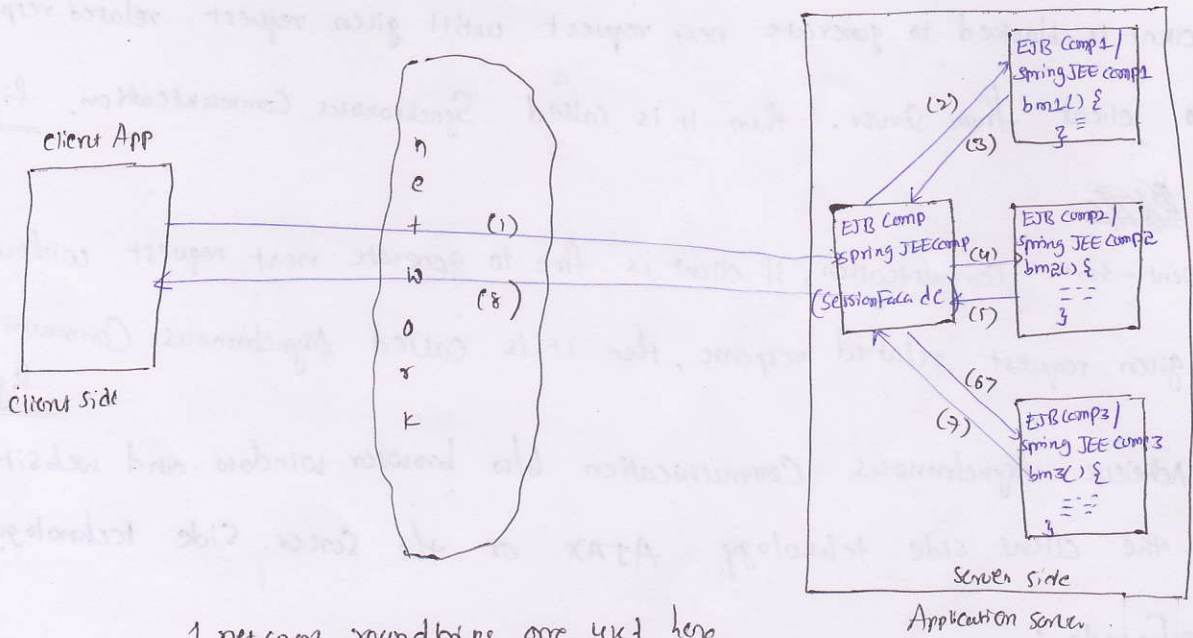
① Session Facade DesignPattern :-

①



3 network roundtrips are used here

②



1 network roundtrips are used here.

Problem:- If Remote client directly talks with multiple business Components there is a chance of getting multiple network round trips b/w client and Business Component as shown above (I) diagram

Solution:- To reduce the network round trips the above (I) scenario take one dummy Business Component at Server Side to receive request from client and make that Component interacting with other Business Component as shown above (II). here the dummy business Component is called as 'Session Facade'. Session Facade doesn't contain any serious business logic. but it reduce

network round trips. when client wants to interacting with multiple Business Component.

→ In the above (II) diagram client application talking with multiple Business Component by just utilizing only ~~one~~ one network round trip.

→ for related info on session facade refer page no's 21 and 22.

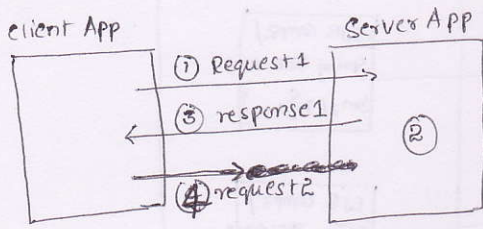


Fig-1

Synchronous Communication b/w client Application and server Application

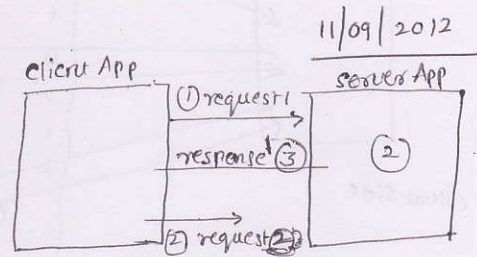


Fig-2

Asynchronous Communication b/w client Application & server Application.

→ If client is blocked to generate next request until given request related response comes back to client from server. then it is called Synchronous Communication, Fig-I

~~XXXX~~

→ In client-server communication, if client is free to generate next request without waiting for given request related response, then it is called Asynchronous Communication, Fig-II

→ To Achieve Asynchronous Communication b/w browser window and website use either the client side technology AJAX or the server side technology portlets.

\* Message Facade :-

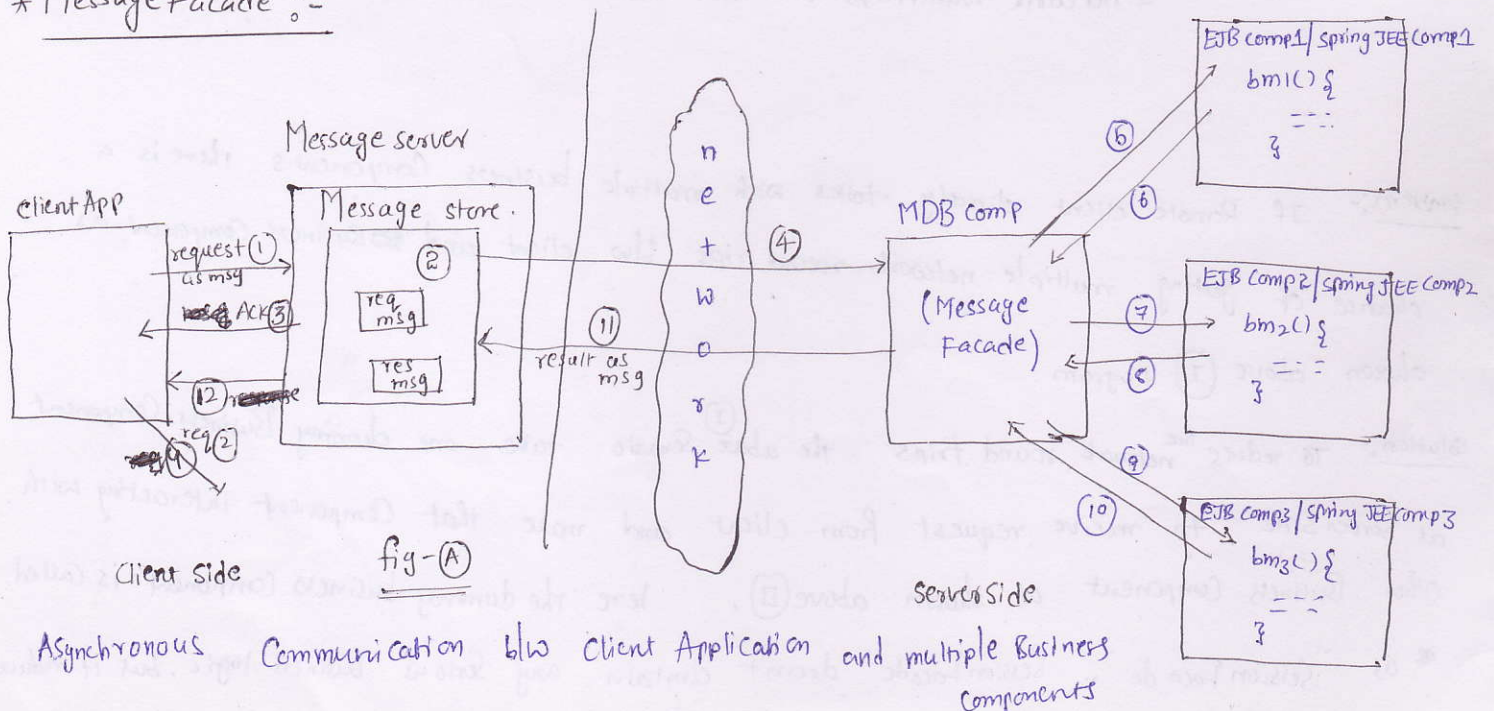


Fig-(A)

Asynchronous Communication b/w Client Application and multiple Business Components



Problem:- The implementation of Session Facade design pattern reduces network roundtrips b/w client and business components, but that communication is synchronous communication. So, the client has to wait to generate next request until the result is received from multiple business components.

Solution:- use JMS and MDB support to make communication b/w client and business components more asynchronous as shown above diagram fig-(A)

For related information on Message Facade design pattern, refer page no:- 22, 23

## DAO : (Data Access Object) :-

Problem:- Mixing up persistence logic with other logics of Application (Specially with business logic) does not give flexibility of modification for persistence logic when database s/w is changed or its details are changed.

Solution:- Implement DAO design pattern. It is a java class that separates persistence logic from other logics of the application development and provides flexibility of modification.

DAO class contains the following logics.

- ① Logic to establish the connection
- ② Logic to release the connection
- ③ Logic to perform CRUD operations on table as per project requirement

NOTE:- In one project we can have either one DAO class (or) multiple DAO classes.

NOTE:- To develop persistence logics in DAO class we can use any persistence technology like jdbc, hibernate & etc....

## \* DAO Factory :-

It is extension of DAO, Factory, patterns which contains the ability to return one database s/w specific DAO class object based on the data that is supplied. This is useful when project deals with multiple database softwares performing some persistence operations on multiple database softwares.

```
public class MyDAOFactory
```

```
{
```

```
// Factory method
```

```
public static DAO getDAO (String name )
```

```
{
```

```
if (name . equals ( "oracle" ) )
```

```
return OracleDAO ( );
```

```
else if ( name . equals ( "Mysql" ) )
```

```
return MysqlDAO ( );
```

```
else
```

```
return null ;
```

```
}
```

```
}
```

```
abstract class DAO
```

```
{
```

```
public abstract void makeConnection ( );
```

```
public abstract void releaseConnection ( );
```

```
public abstract int insertInfo ( -, -, -, - );
```

```
public abstract int updateInfo ( -, -, -, - );
```

```
}
```

```
// DAO class for oracle
```

```
class OracleDAO extends DAO
```

```
{
```

```
public void makeConnection ( )
```

```
{
```

```
----- //logic to establish connection with oracle db/s/w.
```

```
}
```

```
public void releaseConnection ( )
```

```
{
```

```
--- //logic to release connection with oracle db/s/w.
```

```
}
```

```
public int insertInfo ( -, -, -, - )
```

```
{
```

```
--- //logic to insert record into Oracle table
```

```
}
```

```
public int updateInfo ( -, -, -, - )
```

```
{
```

```
--- //logic to update record in Oracle table.
```

```
}
```

```
}
```



// DAO class for MySQL

```
class MySQLDAO extends DAO
```

```
{
```

```
public void makeConnection()
```

```
{  
    --- logic to establish connection with MySQL db slw.  
}
```

```
}
```

```
public void releaseConnection()
```

```
{  
    --- logic to release connection with MySQL db slw
```

```
}
```

```
public int insertInfo (-, -, -, -)
```

```
{
```

```
    --- logic to insert record into MySQL db slw
```

```
}
```

```
public int updateInfo (-, -, -, -)
```

```
{
```

```
    --- logic to update record in MySQL db slw.
```

```
}
```

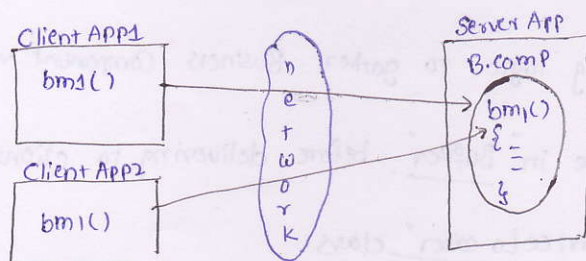
```
}
```

→ DAO Factory is a factory returning one or other DAO class object based on the data that is supplied. This Factory class will be used in other resources of the project to get their choice DAO class object and to perform persistence operations by using that DAO class object.

Business Delegate :-

12/09/2012

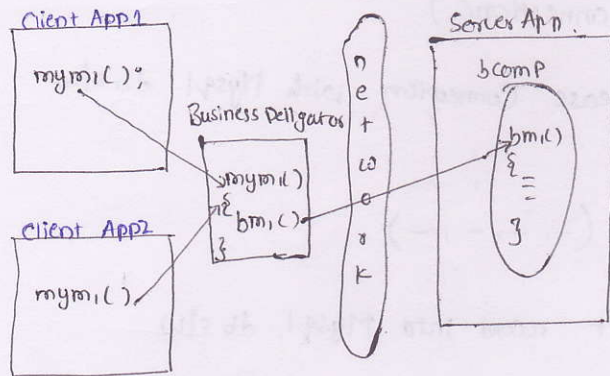
Problem :- Business In client applications calls business methods of business components available in Server application directly we don't get flexibility of modification if business method details are changed in future.



Solution :- Developed helper class for Client Application having the logics to call the business methods of Server Application. so, Any modifications are there in the details of

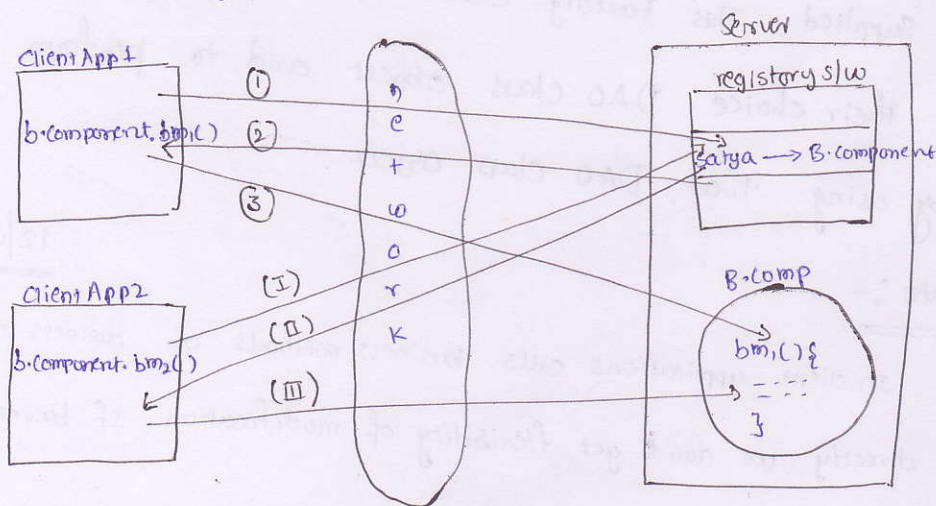
Business method can perform them Business Delegation class and there is no necessity of Disturbing client Applications.

→ In one angle it calls struts ActionClass as Business Delegate when it is having logic calling the business methods of Model Layer EJB components (or) Spring applications.



### Service Locator :-

Problem :- To call Business methods of Business Components Belonging to Server the client Applications must gather Business Component references from registry/sw using JNDI code. If Every client application separately gathers this Business Component reference from registry that <sup>may</sup> increase <sup>more</sup> ~~many~~ network round trips.



Solution :- Develop Helper class having logic to gather Business Component reference from registry of server and to keep that one in Buffer before delivering to client Applications.

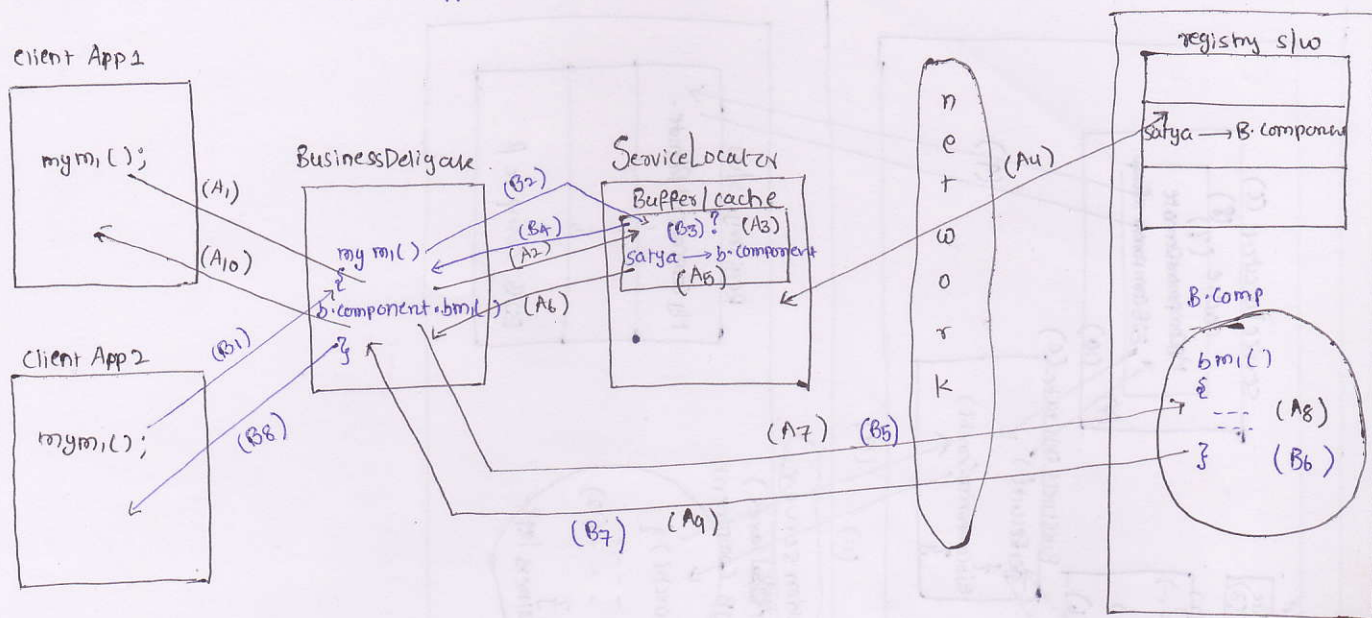
This helper class is called "Service Locator" class.

→ we generally develop this Service Locator class as Singleton Java class to avoid multiple buffers to creations.



We generally use ServiceLocator class along with Business Delegate class.

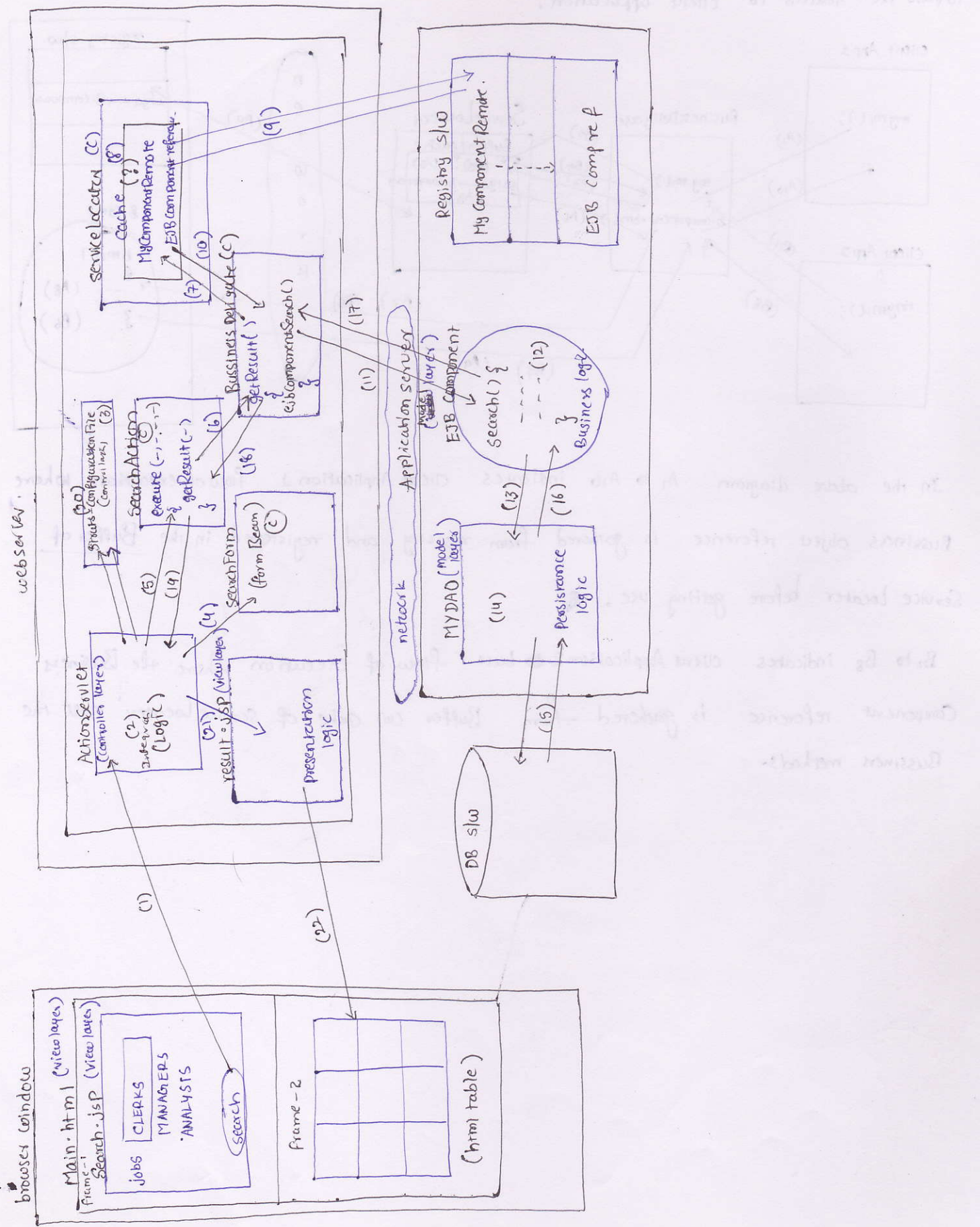
→ here ServiceLocator responsible to gather and give business object / component reference and Business Delegate responsible to call Business methods by using Business object reference and to pass the results to client application.



In the above diagram A<sub>1</sub> to A<sub>10</sub> indicates client Application 1 flow of execution where Business object reference is gathered from registry and registered in the Buffer of Service Locator before getting use.

B<sub>1</sub> to B<sub>8</sub> indicates client Application 2 flow of execution where the Business Component reference is gathered from Buffer (or) cache of service locator call the Business methods.

# MINI Project by using Struts, EJB and JDBC technologies having the implementation of multiple Design Patterns :-





- ① Form Page submits the request by selecting one item of select box
- ② As a Controller ActionServlet traps and takes the request
- ③ ActionServlet uses the entries of Struts-configuration file decide Formbean and Actionclass to process the request.
- ④ ActionServlet writes the formdata to FormBean class object.
- ⑤ ActionServlet calls the execute() of ActionClass.
- ⑥ This execute() passes the request to BusinessDeligate class
- ⑦ & ⑧ BusinessDeligate uses ServiceLocator class and checks the availability of EJB Component reference in Buffer (or) cache
- ⑨ Since EJB Component reference is not available in Buffer it will be gathered from the registry slw of Application Server and will be registered with the cache (or) Buffer of ServiceLocator.
- ⑩ ServiceLocator passes the EJB component reference to Business Deligate class
- ⑪ BusinessDeligate class uses that EJB component reference and calls the Business method of EJB Component.
- ⑫ & ⑬ & ⑭ & ⑮ & ⑯ This Business method takes the Support of DAO class to send and execute SQL select query DataBase table and gets the records into ArrayList object.
- ⑰ Business method of EJB Component passes the result (ArrayList object) to Business Deligate class
- ⑱ Business Deligate class passes the result to execute()
- ⑲ execute() keep the result in request attribute and transfer the controle to ActionServlet
- ⑳ & ㉑ ActionServlet uses the entries of Struts-configuration file get the result page and pass the control to Result.jsp.
- ㉒ Result.jsp formats the results by using presentation logic and sends that result to 2<sup>nd</sup> frame of webpage as HTML table content.

## Explicitly implemented Design patterns of the above project :-

- ① MVC R M - ejb component V - jsp programs C - ActionServlet
- ② DAO
- ③ D.T.O class / V.O class
- ④ Business Delegate
- ⑤ Service Locator
- ⑥ Singleton java class } required while implementing ServiceLocator.
- ⑦ Factory method }
- ⑧ View Helper Design Pattern

## Some Implicit Design Patterns of the above project :-

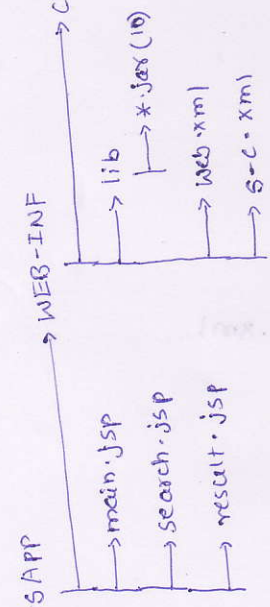
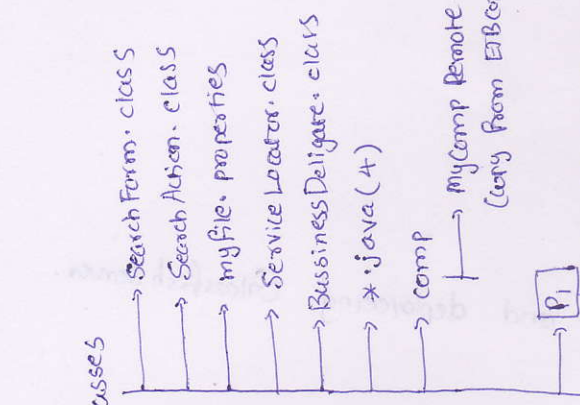
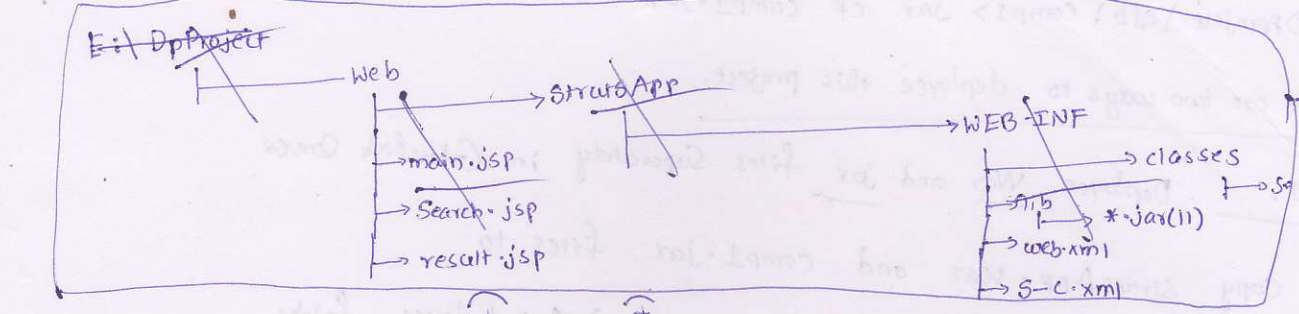
- ① Front Controller ( ActionServlet )
- ② Dependency Injection / IOC (~~Form Data is written to Form~~  
(The way ActionServlet writes FormData to FormBean class obj))
- ③ Abstract Controller (The pre-defined helper class to ActionServlet which is nothing but Request processor or class)
- ④ D.T.O class / V.O class (FormBean class)
- ⑤ Template Method (The process (-,-) of Request Processor)

For the above diagram base MiniProject having the implementation of Design Pattern.

refer the Supplementary HandOut given on 13/09/2012



Procedure to deploy and execute design project based mini project of handout



E:\DpProject  
└── web

jar files in classPath :-

servlet-api.jar  
struts-core-1.3.8.jar

jar files in WEB-INF\lib folder  
10 regular Struts jar files

prepare war file representing struts Application  
e:\DpProject\web\strutsApp > jar cf strutsApp.war

E:\DpProject



jar files in class Path  
javaee.jar  
c0dbcl4.jar

jar files in C:\classPath-home > AppServer\domains\mydomain\lib\ext

Prepare jar files representing EJB Component :-

E:\DpProject\ejb\Comp1 > jar cf Comp1.jar

there are two ways to deploy this project.

Approach 1:- Deploy War and jar files Separately in Glassfish Server

Copy StrutsApp.war and comp1.jar files to

<Glassfish-home>\Appserver\domains\mydomain1\Autodeployed folder

Request URL: HTTP://localhost:8080/myweb

Open browser window

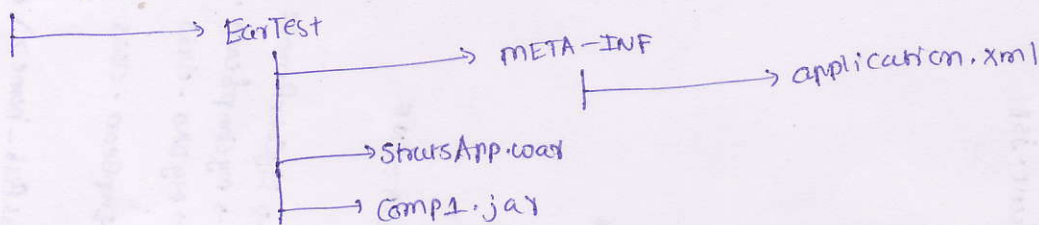
Approach 2:- Combine War and Jar files Ear file and deploying Glassfish server.

ear file = jar file + jar file + . . . . .

ear file = jar file + war file + . . . . .

ear file = war file + war file + . . . . .

E:\DpProject



to prepared ear file

E:\DpProject\EarTest > jar cf FinalApp.ear

Application.xml  $\leftarrow$  DD File For Ear file

<doctype ! application PUBLIC  
//EN "http://java.sun.com/combined:application-1-3.dtd" >

```

<application >
  <display-name > earTest </display-name >
  <description > Struts EJB project </description >
  <module >
    <ejb > Comp1.jar </ejb >
  </module >

```



```

1 =====
2 App2 http://localhost:8080/strutsApp -> (1)
3 =====
4 -----main.jsp----- welcome page (v) -> (3)
5 <frameset rows="30%,70%">
6   <frame name="f1" src="Search.jsp">
7   <frame name="f2"/>
8 </frameset>
9 -----Serach.jsp----- FormBased (view)
10 <%@taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
11 <%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
12 <html:form action="searchpath" target="f2"> x1 (ref 7)
13   (5) -> Action Path of Action class (refer line no: 69)
14   <bean:message key="my.lbl"/>
15   <html:select property="job" multiple="yes">
16     <html:option value="CLERK">CLERKS</html:option>
17     <html:option value="MANAGER">MANAGERS</html:option>
18     <html:option value="ANALYST">ANALYSTS</html:option>
19     <html:option value="SALESMAN">SALESMEN</html:option>
20   </html:select>
21
22   <html:submit>
23     <bean:message key="btn.cap"/> }
24   </html:submit> }
25
26
27 </html:form>
28 -----myfile.properties-----
29 # To change this template, choose Tools | Templates
30 # and open the template in the editor.
31 my.lbl=Select job(s)
32 btn.cap=Search
33 -----web.xml-----
34 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
35   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
36   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
37     http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
38   <servlet>
39     <servlet-name>action</servlet-name>
40     <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
41     <init-param>
42       <param-name>config</param-name>
43       <param-value>/WEB-INF/struts-config.xml</param-value>
44     </init-param>
45     <load-on-startup>2</load-on-startup>
46   </servlet>
47
48   <servlet-mapping>
49     <servlet-name>action</servlet-name>
50     <url-pattern>*.do</url-pattern>
51   </servlet-mapping>
52
53   <welcome-file-list>
54     <welcome-file>Main.jsp</welcome-file> (2)
55   </welcome-file-list>
56 </web-app>
57 -----struts-cogif.xml
58 <!DOCTYPE struts-config PUBLIC
59   "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
60   "http://jakarta.apache.org/struts/dtds/struts-config_1_3.dtd">
61
62 <struts-config>
63   <form-beans>
64     <form-bean name="sf" type="SearchForm"/> (8)
65   </form-beans>
66
67
68   <action-mappings>
69     <action name="sf" path="/searchpath" type="SearchAction">

```

Select box allowing us to select multiple items at a time

x2) when form is submitted to web application the output Generated by the web resource program of webApp will be displayed in frame whose name is "f2"

Deployment Descriptor File

38-51 (6)

struts Configuration File (Controller)

Formbean Configuration

69 to 77 Struts Action class cfg (7)



```

70 <forward name="success" path="/result.jsp"/>
71 </action>
72 </action-mappings>
73
74 <message-resources parameter="myfile"/>
75
76 </struts-config>

```

```

77 -----SearchForm.java----- FormBeans (Controller layer)
78 import javax.servlet.http.HttpServletRequest;
79 import org.apache.struts.action.ActionMapping;
80 import org.apache.struts.action.ActionForm;
81

```

```

82 public class SearchForm extends ActionForm {
83     private String job[];
84     public void reset(ActionMapping mapping, HttpServletRequest req)
85     {
86         job=new String[1];
87         job[0]="no item selected";
88     }
89     public String[] getJob() {
90         return job;
91     }
92     public void setJob(String[] job) {
93         this.job = job;
94     }
95 }

```

```

96 -----SearchAction.java----- Action Class (Controller layer)
97 import java.util.ArrayList;
98 import javax.servlet.http.*;
99 import org.apache.struts.action.*;
100

```

```

101 public class SearchAction extends Action {
102     @Override
103     public ActionForward execute(ActionMapping mapping, ActionForm form,
104     HttpServletRequest request, HttpServletResponse response)
105     throws Exception {
106         //read form data
107         SearchForm sf=(SearchForm)form;
108         String jobs[]=sf.getJob();
109         System.out.println("In execute(-,-,-):SearchAction");
110         System.out.println("before calling B.method");
111     }
112     // call B.method
113     BusinessDelegate bd=new BusinessDelegate();
114     ArrayList al=bd.getResult(jobs);
115     System.out.println("after calling B.method");
116 }

```

```

117 // send result to result page as req attribute
118 request.setAttribute("result",al);
119 //forward control to result page
120 return mapping.findForward("success");
121 } //execute(-,-,-)
122 } //class

```

```

123 -----ServiceLocator.java----- Class implementing Service Locator Design Pattern
124 import java.util.*;
125 import javax.naming.*;
126

```

```

127 public class ServiceLocator
128 {
129     private Hashtable Cache;
130     private InitialContext ic;
131     private static ServiceLocator sl=null;

```

9

10

13

25

26

logic to read form data from FormBean class object

to hold multiple values collected from list box

logic to handle no item selected state of list box while working with session scoped FormBean

Action Class (Controller layer)

calling a method of Business Delegate class gather the result from EJB component Business method (refer line no: 196 to 206)

Class implementing Service Locator Design Pattern Control Layer

It is Singleton class  
To maintain local cache or buffer having EJB component reference gathered from registry.  
To maintain connectivity with registry server.

to supporting Singleton



```

139 private ServiceLocator() // private constructor (to support Singleton)
140 {
141     try
142     {
143         Cache = new Hashtable();
144
145         // jndi properties
146         Hashtable ht=new Hashtable();
147         ht.put(Context.INITIAL_CONTEXT_FACTORY,
148             "com.sun.enterprise.naming.SerialInitContextFactory");
149         ht.put(Context.PROVIDER_URL,"iiop://localhost:4848");
150         ic=new InitialContext(ht);
151     } // try
152     catch(Exception e)
153     {
154         e.printStackTrace();
155     }
156 } // constructor

```

Logic to Establish connection with GlassFish registry by using its JNDI properties

ServiceLocator contains one member variable "cache"  
Cache contains Buffer

```

159 // Factory method having singleton logic
160 public static ServiceLocator getLocator() // factory method.
161 {
162     if(sl==null)
163         sl=new ServiceLocator();
164
165     return sl;
166 } // getLocator()

```

16

```

167 // method having ServiceLocator Impl logic
168 public Object getService(String jndiName) throws Exception

```

```

169 {
170     System.out.println(" ServiceLocator:getService(-) ");
171     try
172     {
173         if(!Cache.containsKey(jndiName))
174             Cache.put(jndiName, ic.lookup(jndiName));
175         } // if
176     } // try
177     catch(Exception e)
178     {
179         e.printStackTrace();
180     } // catch

```

18

if given JNDI name is not available as key in local cache then EJB component reference will be gathered from registry based on JNDI name and it will be placed in local cache along with JNDI name.

```

181 // Return object from cache
182 return Cache.get(jndiName);

```

The EJB component reference of given JNDI name will be returned by gathering it from local cache.

```

183 } // getService()
184 } // class
185 -----BusinessDelegate.java----- class implementing Business Delegate Design Pattern (Control layer)
186 import java.util.*;
187 import comp.MyCompRemote;

```

```

188 public class BusinessDelegate
189 {
190     public ArrayList getResult(String job[]) throws Exception
191     {
192         System.out.println("getResult method of BusinessDeligate class");

```

14

15 ServiceLocator sl= ServiceLocator.getLocator(); Use the singleton object of ServiceLocator.  
Object obj =sl.getService("comp.MyCompRemote"); logic to get EJB component reference from the cache maintained by ServiceLocator

17

```

193 // Calling B.method
194 ArrayList al=wor.search(job);
195 return al;

```

19 calls the bussiness method of EJB component

24

returns to execute method of Actionclass.

```

196 }

```



```

208 -----result.jsp-----
209 <%@page import="java.util.*,java.math.*,p1.EmpBean" %>
210
211 <%
212     ArrayList al=(ArrayList)request.getAttribute("result");
213 %>
214 <center>
215     <table border width="100%" bgcolor="#FFFFFF">
216     <tr>
217         <th>EMPNO</th>
218         <th>ENAME</th>
219         <th>JOB</th>
220         <th>SAL</th>
221     </tr>
222
223 <%
224     for(int i=0;i<al.size();i++)
225     {
226         EmpBean eb=(EmpBean)al.get(i);
227         BigDecimal bd=new BigDecimal(eb.getSal());
228     %>
229     <tr>
230         <td><b><%=eb.getEmpno() %></b></td>
231         <td><b><%=eb.getEname() %></b></td>
232         <td><b><%=eb.getJob() %></b></td>
233         <td><b><%=bd.setScale(2,BigDecimal.ROUND_HALF_UP)%></b></td>
234     </tr>
235 <%
236     }
237 %>
238 </table>
239 </center>
240
241
242 <%--<%@taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
243 <%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
244 <%@page import="java.util.*,p1.EmpBean"%>
245
246 <% System.out.println("In result.jsp"); %>
247
248 <table border=1>
249 <%
250 ArrayList al=(ArrayList)request.getAttribute("result"); → reads the request attribute value
251 %>
252
253 <logic:notEmpty name="result" scope="request"> → checks whether the request attribute is empty or not.
254     <tr>
255         <td>ENO</td>
256         <td>ENAME</td>
257         <td>DESG</td>
258         <td>SALARY</td>
259     </tr>
260
261     <logic:iterate id="id1" collection="<%=al%>" →
262     <tr>
263         <td> <bean:write name="id1" property="empno" /></td>
264         <td> <bean:write name="id1" property="ename" /></td>
265         <td><bean:write name="id1" property="job" /></td>
266         <td><bean:write name="id1" property="sal" /></td>
267     </tr>
268 </logic:iterate>
269 </logic:notEmpty>
270 </table> --%>

```

logic to go through the elements of ArrayList, in each iteration 'id1' points to one element of ArrayList i.e. EmpBean class object

logic to display the records of ArrayList as HTML table content



1 =====  
2 App1  
3 =====

4 -----MyCompRemote----- Business Interface of EJBComponent (Model layer)  
5 package comp;  
6 import java.util.ArrayList;  
7 import javax.ejb.Remote;

8  
9 @Remote → makes Business interface has Remote Business Interface.  
10 public interface MyCompRemote {  
11 public ArrayList search(String job[]);  
12 }

13 -----MyCompBean.java (bean class of EJB component) (Model layer).  
14 package comp;  
15 import java.util.ArrayList;  
16 import javax.ejb.Stateless;  
17 import p1.EmpBean;

18  
19 @Stateless → makes the Bean as SLST (Stateless session Bean)  
20 public class MyCompBean implements MyCompRemote {

21  
22 public ArrayList search(String[] jobs) {  
23 System.out.println("MyCompBean:search(-) method");  
24  
25 // write b.logic to frame condition required for sql query  
26 StringBuffer sb=new StringBuffer();  
27 sb.append("(");  
28 for(int i=0;i<jobs.length;i++)  
29 {  
30 if(i == jobs.length-1)  
31 sb.append("'+jobs[i]+'");  
32 else  
33 sb.append("'+jobs[i]+' ,");  
34 }  
35 sb.append(")");  
36 String cond=sb.toString(); → gives condition required for the sql query like ('CLERK', 'MANAGER')

job[]	
CLERK	0
MANAGER	1

37  
38 // use the persistence logic of DAO class  
39 MyDAO dao=new MyDAO();  
40 ArrayList al=dao.findEmployees(cond); → refer line no's 54 to 93  
41 return al;  
42 } // search → returns to Business Delegate class.  
43

refer line no's 54 to 93

44 } // class  
45 -----MyDAO.java (Model layer)

46 package comp;  
47 import java.sql.\*;  
48 import java.util.\*;  
49 import p1.EmpBean;

50  
51 public class MyDAO {

52  
53  
54 public ArrayList findEmployees(String cond)  
55 {  
56 // method having persistence logic  
57 System.out.println("findEmployees(-):MyDAO");

54-93  
22

58  
59 Connection con=null;  
60 Statement st=null;  
61 ResultSet rs=null;  
62 ArrayList al=new ArrayList();  
63  
64 try  
65 {  
66 Class.forName("oracle.jdbc.driver.OracleDriver");  
67 con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:satya","scott","ti");  
68 st=con.createStatement();  
69 rs=st.executeQuery("select empno,ename,job,sal from emp where job in "

Page 1 of 2  
select empno, ename, job, sal from emp where job in ('CLERK', 'MANAGER') order by job

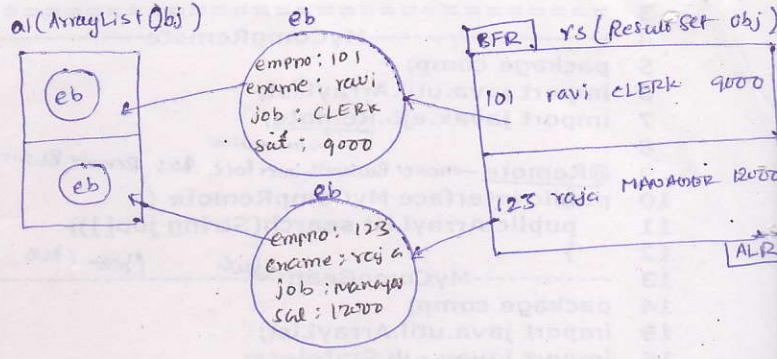


logic to transfer the records of ResultSet object to ArrayList by taking the help of D.T.O class

```

70         + cond+" order by job");
71     while(rs.next())
72     {
73         System.out.println("In while");
74         EmpBean eb=new EmpBean();
75
76         eb.setEmpno(rs.getInt(1));
77         eb.setName(rs.getString(2));
78         eb.setJob(rs.getString(3));
79         eb.setSal(rs.getFloat(4));
80
81         al.add(eb);
82     } //while
83     rs.close();
84     st.close();
85     con.close();
86 } //try
87 catch(Exception ee)
88 {
89     System.out.println(ee.toString());
90 }
91
92     return al;
93 } //findEmployees
94 } //class

```



```

95 -----D.T.O class / V.O. class-----
96 -----EmpBean-----
97 package p1;
98 import java.io.*;
99
100 public class EmpBean implements Serializable
101 {
102     private int empno;
103     private String ename;
104     private String job;
105     private float sal;
106
107     public int getEmpno() {
108         return empno;
109     }
110
111     public void setEmpno(int empno) {
112         this.empno = empno;
113     }
114
115     public String getEname() {
116         return ename;
117     }
118
119     public void setName(String ename) {
120         this.ename = ename;
121     }
122
123     public String getJob() {
124         return job;
125     }
126
127     public void setJob(String job) {
128         this.job = job;
129     }
130
131     public float getSal() {
132         return sal;
133     }
134     public void setSal(float sal) {
135         this.sal = sal;
136     }
137 }

```

When we deploy EJB 3.x Component in GlassFish server when the Business interface name of that Component automatically becomes JNDI name but EJB Component reference that is placed in the registry. For the above Component Comp.MyCompRemote is the Jndi name.



I would try to update our site [JavaEra.com](http://JavaEra.com) everyday with various interesting facts, scenarios and interview questions. Keep visiting regularly.....

**Thanks and I wish all the readers all the best in the interviews.**

# [www.JavaEra.com](http://www.JavaEra.com)

A Perfect Place for All **Java** Resources