www.tutorial4us.com

Tutorial4Us

College Projects

```
void main()
{
 int no;
 clrscr();
 printf("Hello");
}
```

Basic Program

C Tutorial

JAVA Programming

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

30/08/2012

Design pattern :-

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

Design patterns are best practices to use s/w Technologies effectively in project of application development.

→ Design pattern are Supporting code while developing s/w projects by using plain technologies.

→ The worst solution for 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 Complition.

→ There are 500+ design patterns in Java Environment.

→ Design patterns can be implemented by using many s/w technology. (or) programing 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 (Inversion of control) pattern

Adapter class

Fastline reader

~~DO class object / value class object~~

V.O class object / 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 Filter

Abstract Controller

-: Integration Layer Design patterns :-

SessionFacade
MessageFacade
ServiceLocater
BusinessDeligate

→ Senior developers designs solutions for reoccuring 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 wasting memory, degrading performance. It is recomended to create one object and use it for multiple times.

Solution :- Use Singleton java class.

The java class that allows us 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 locater class will be implemented as Singleton java class.

Rules to implement Singleton java class :-

① it must have only private Constructors.
② must have private Static reference Variable of same class
③ override clone() to suppress cloning process.
④ must have public 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
{
   //satic reference variable
   private static STest n=null
   // private Constructor
   private STest()
   {
     S.O.P("STest : 0-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 ob " +t4.hashcode();
  }
}
```

> javac SingleTonTest.java

> java SingletonTest.

31/08/2012

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

② Synchronized Singleton java class :-

when multiple threads acts 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 recomended 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();
Calender  cl = Calender.getInstance();
```

↳ it is not calender class object. it is object of gregorian Calender class which is subclass of Calender class (abstract class).

→ Static factory methods are Useful 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 recomended

Examples on instance factory method :-

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

```
StringBuffer sb = new StringBuffer("hello");
String s2 = sb.substring(0,3); // hell
```

method of String class returns String class obj

```
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 factory methods returning empty objects (objects containing default data)

Application:-

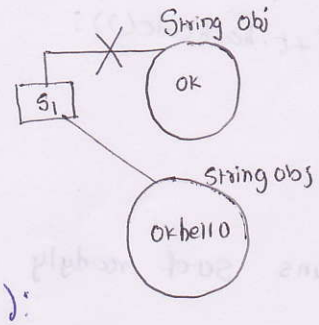
For example code on Factory pattern / Factory method. refer page no 1 & 2

Prototype of InstanceFactory method :-

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

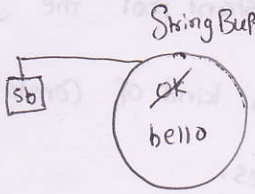
```
String s1 = new String("ok");
String s1 = s1.concat("hello"); // okhello
```

(String class is immutable class)




```
StringBuffer sb = new StringBuffer("ok");
sb.append("hello");
```

(StringBuffer class is Mutable class)



most of the predefined classes & programer 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 modify(Data~~

  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("t object hashcode:"+t.hashcode());
    t.modifyData();
    t.disp();
    s.o.p("t 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).

→ 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 (It's class is called as immutable class).

01/09/2012

(4) String Constant Pool :-

→ String Constant pool maintains set of readyly 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 develope 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.

```
//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,"rajaa");
  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();
}//main
}//class
```

NOTE:- only final classes are not Immutable cla

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

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

(5)

*) Template method Design pattern :-

problem :-

```
task1 ---> 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();
}

task1 : mymethod();
```

now only one method needs to be call to complete task.

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

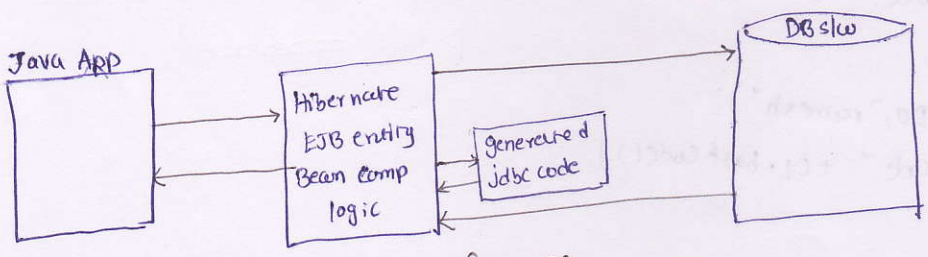
So, this method is called Template method.

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

(6)

*) FastLine Reader :-

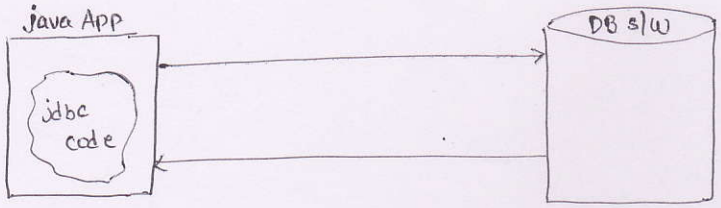
problem :- using high and ORM s/w's (like hibernate) to develop persistence logic in our small and medium scale apps may degrade the performance because these ORM s/ws internally generate jdbc code to complete the persistence logic development



may degrade the performance.

Solution :- Use FastLine Reader design pattern

Write jdbc code directly in our small and medium Scale applications to interact with the db s/w 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:odbc:oradsn", "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 classes 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 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 page nos: 2

Prototype :-

Problem :- Creating new object from scratch level having the existing object data has initial data is ~~set~~ quite time consuming process. ~~actually required~~

Solution :- Use cloning to create new object where new object contains existing object 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 their is no need of objects initializing new objects that are created to cloning (as they old existing object 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 Exceptions
  {
    //create first object
    Demo d1 = new Demo(10,20);
    S.O.P("d1 hashcode:" + d1.hashCode());
    d1.disp();
      //perform cloning
    Demo d2 = (Demo)d1.myClone();
    S.O.P("d2 hashcode:" + d2.hashCode());
    d2.disp();
  } //main
} //class
```

backup > NAPPs > APPs > Design pattern > prototype > javac CloneableTest.java

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

→ 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 Designed pattern refer Page no's 11 & 12

(9) 05/09/2012

Builder Design pattern:-

problem:- Construction of Complex objects directly by taking one Complex class is not recomended 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()
   {
     xtarn 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 Frieg().price() + new Drink().price();
  }
}
```

```
public class BuilderTest
{
    p s v m(String[] args)
    {
        //complex object 1
MealBuider mb=new MealBuider();
int val=mb.calcPrice();
S.O.P("the meal price is "+val);

        //complex object 2
SnackBuilder sb=new SnackBuider();
int val1=sb.calcPrice();
S.O.P("the snack price is "+val1);
    }
}
```

backup> NAPPS> APPS> DesignPattern> BuilderPattern

```
/> javac BuilderTest.java
/> java BuilderTest
```

→ The ActionMapping object of Struts 1.x environment is created based on Builder Design Pattern.

→ The request, response objects of Servlet Programing will be created by Container based on Builder Design Pattern.

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

Adapter class :-

problem :- when class of the application directly implements interface it as provide implementation for all the ~~empty~~ methods of that interface even though it is not interested to provide implementation 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 intrested 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 method definitions for interface methods. so the class of Application can override its own choice method by extending from Adapter class.

```
class (abstract) 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 instansiation for Adapter class.

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

→ Javax.Servlet.GenericServlet class is partially an adapter class for javax.servlet.Servlet (Servlet interface name) (package name)

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

Ans:- PL designs API specification of the project having rules (method declarations) and guidlines (method definitions / Concreate methods) to programer. this process we uses abstract classes Supplies both rules and guideline and we uses interfaces to supply only rules (method declarations).

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

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

→ while developing adapter classes abstract classes will be taken. while developing business Components Service Interfaces (or) are java Interface.

→ provide Special Run-time Capabilities to objet to JRE marker interfaces are required.

→ May ordinary java class as Special compenent 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 instead of ResultSets
(all Rowsets are Serializable objects by default)

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

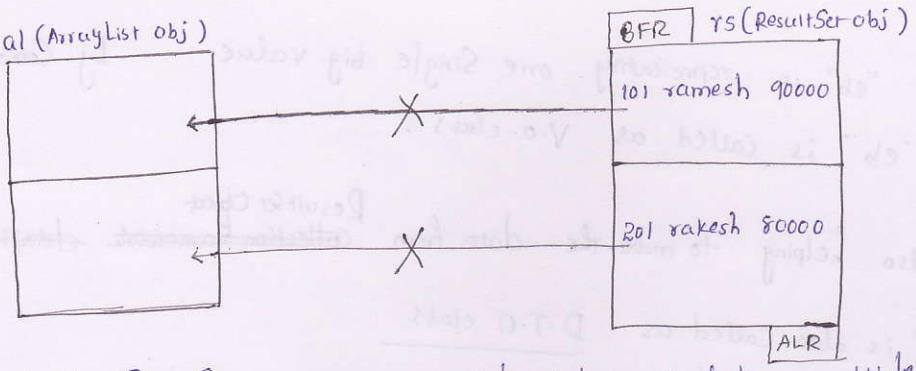
NOTE:- Since most of jdbc driver's are not Supporting Rowsets so use Approach 2 to solve the above problem.

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

→ After moving data to collection framework datastructure if you are looking to perform both read and write Operations Simultaneously then use Synchronized datastructure 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) (DataTransfer Object)

Example Scenario :-

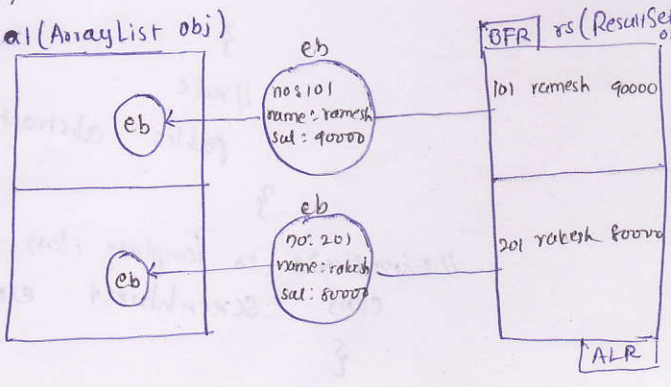
V.O class / D.T.O class (JavaBean)

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.

Simillerly 'eb' is also helping to move the data from ResultSet Object ~~Collection framework~~ ~~datastructure~~ to ArrayList elements. so, the class of 'eb' is also called as D.T.O class

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.

---

(11)

*) Template class :- (DesignPattern)

problem:- Developing multiple classes of Same category from Scratch level kills the reusability and increases the burden on the programers.

Solution:- Use Template class providing mould for programers having rules and guidelines. so, programers 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);
}

//Extention 1 to Template class
class ScreenWriter1 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.valueOff(res);
```

~~BufferedWriter~~ BufferedWriter bw = new ~~BufferedWriter~~ BufferedWriter (new OutputStreamWriter(new FileOutputStream("abc.txt")));

```
      bw.write(s1);
      bw.flush();
      bw.close();
    } //try
    catch (Exception e) { }
 } //catch
} // class


public class TemplateTest
 {
    P S VM (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 :- 07/09/2012

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 metorial 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 code~~.

Dependency Injection:- If underlaying container / Framework / server / Run-time environment dynamically Assigns values to resource then it is calls "dependency Injection"

→ In Dependency Injection underlaying server / Container..... pushes the values to ~~resource~~ resource. so, ~~used~~ 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 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

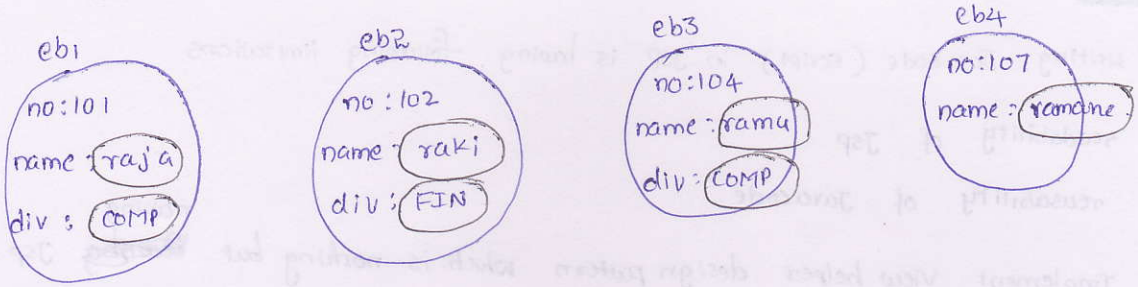
→ The way object is initialize through constructor execution comes under Dependency Injection.

→ The way object is assigned with data through method calls comes under Dependency lookup.

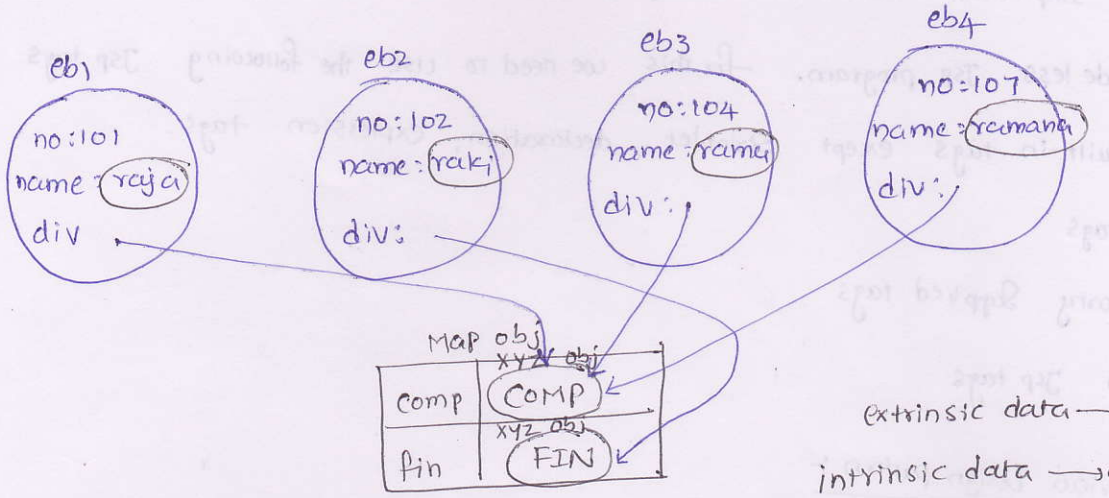
~~(13) Lightweight~~

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 data. 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 supplamentary 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 helper design pattern which is nothing but making ~~creating~~ 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.

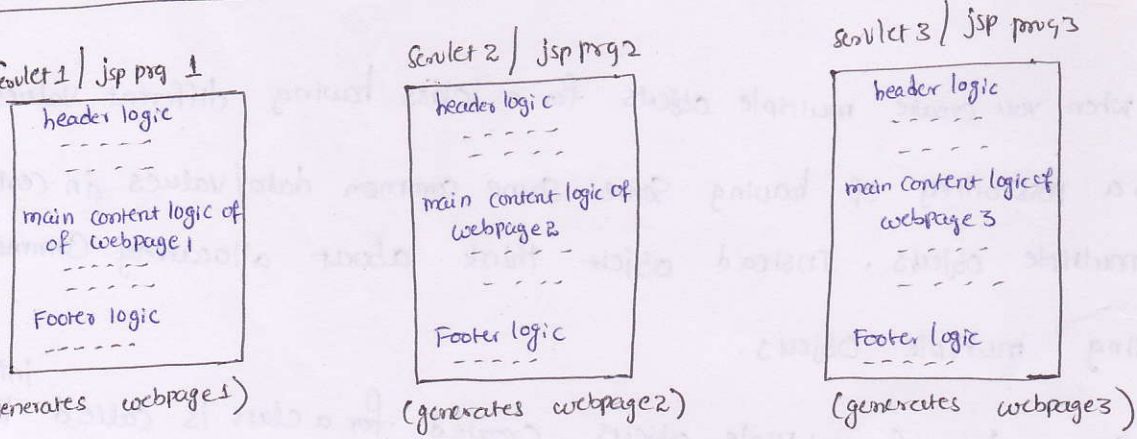
② JSTL tags

③ Third party Supplied tags

④ Custom JSP tags

② Composit View Design pattern :-

Problem :-

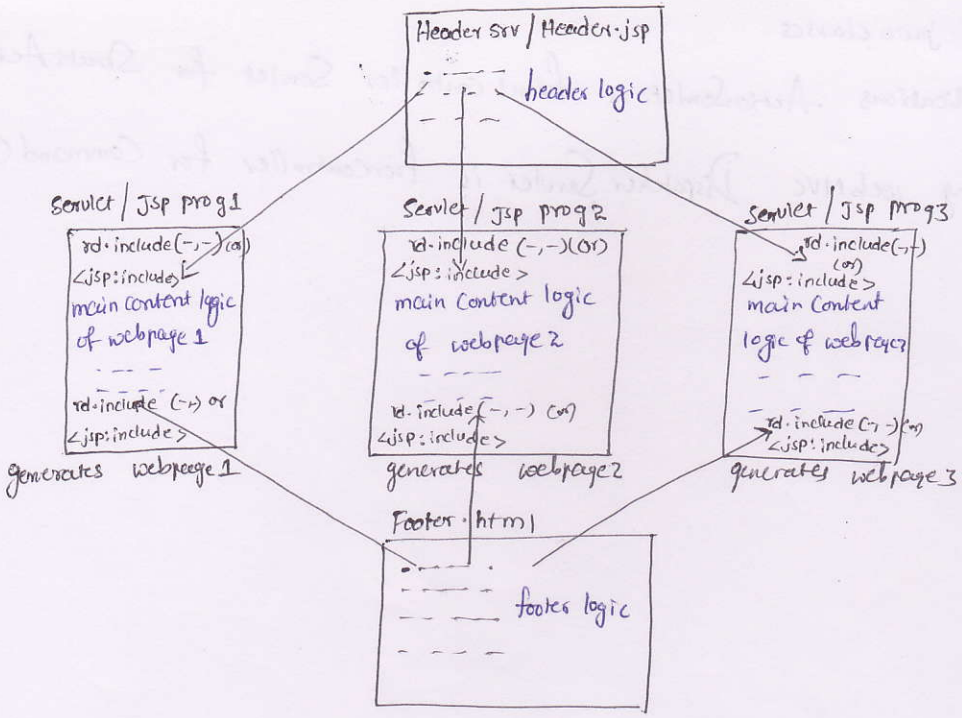


Here the common logic of multiple web resources program are not reusable logic (like header, footer logics) ~~of~~ because they are ~~need~~ hard coaded in every web resource program.

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

Solution :- make the Common logics as reusable logics by keeping them in Separate webresource programs and include their output. ~~to make~~ The main web resource programs its generate webpages for this we need to use rd.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)

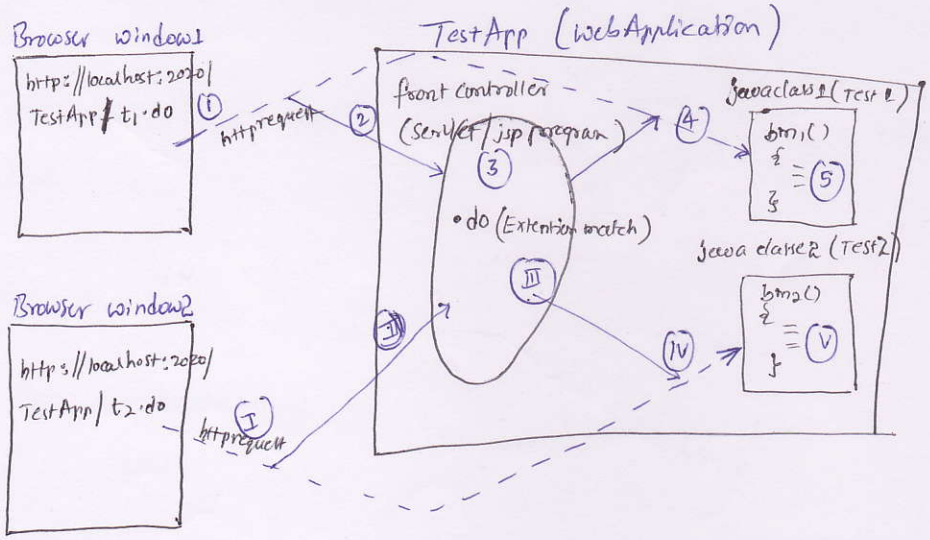
⑤ Front Controller Design pattern :-

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~~ 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 web resource 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 web MVC DispatcherServlet is FrontController for Command Controller classes (Java classes)

web.xml

Configure FrontSrv program in web.xml file with *.do / *.dourl 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) thro
  {
    // 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"))
    {
```

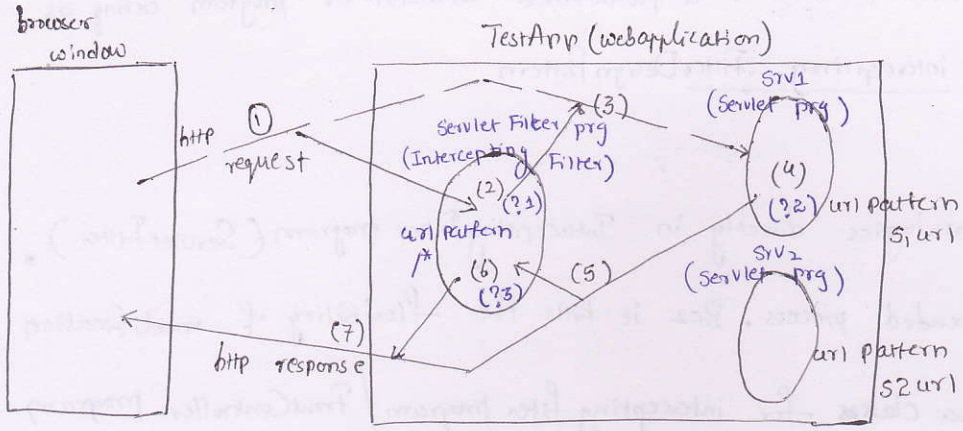
```
Test1 t1 = new Test1();
result = t1.bm1(10,20);
} //if
else if (req.getServletPath().equals("/t2.do"))
{
  Test2 t2 = new Test2();
  result = t2.bm2(10,20);
}
// display the result
pw.println("<br><b> The result is :" + result);
} //doGet()
public void doPost(HttpServletRequest req, HttpServletResponse 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 webresource programs. is called as "Intercepting Filter".

Problem :- keeping the common and global pre-request processing logic and post response generation logic in every main webresource program of webapplication 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 processing logic in ServletFilter program.

① Authentication logic
② Authenization logic
③ logging logic

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

(NOTE 1) :- FrontController 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

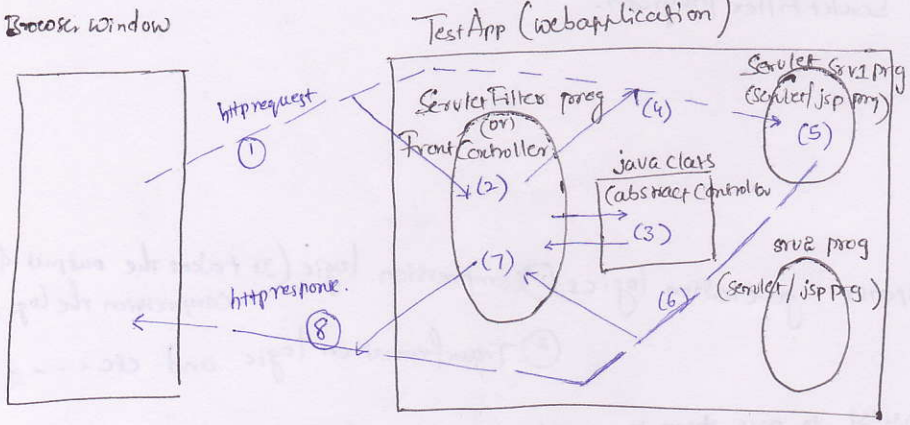
```
public class MyFilter implements Filter
{
  Public void init (FilterConfig fg)
  {
    -----  //initialization logic
  }

  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 coading main logics directly in Intercepting Filter program (ServletFilter), FrontController is not recomended process. Bcz it kills the flexibility of modification

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



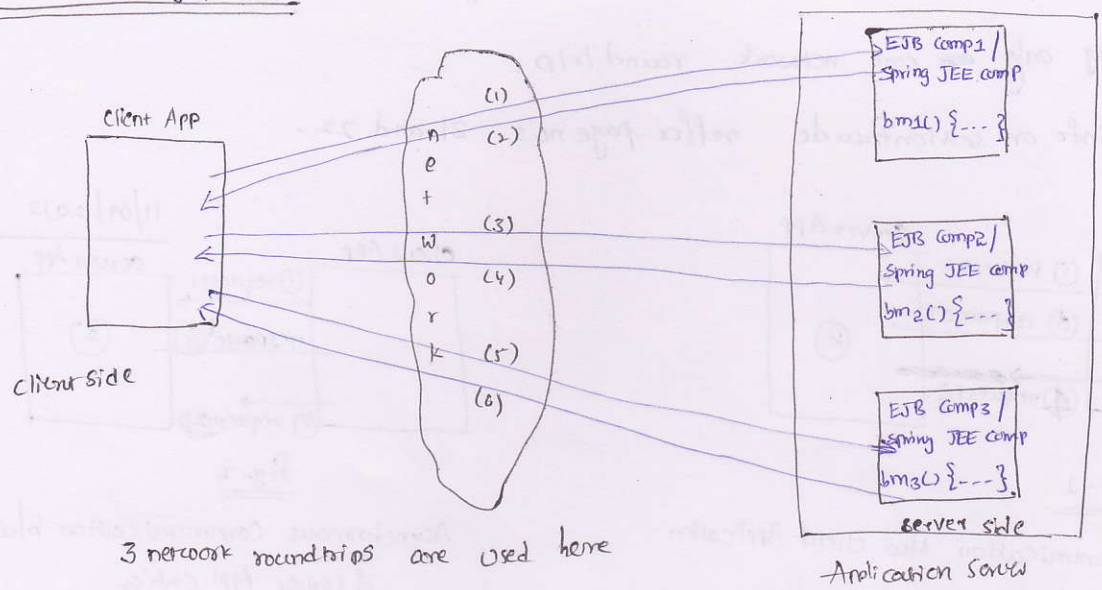
→ Abstract Controller class is very useful to customize the logics of front controller program (or) IntercptingFilter program being from outside of these programs.

→ in Struts 1.x environment RequestProcessor class is Abstract Controller for the Front Controller ActionServlet. Simlerly In Struts 2.x environment Intercepters are the abstract Controllers for IntercptingFilter/ controller called FilterDispatcher.

III) Integration layers Design Pattern *)

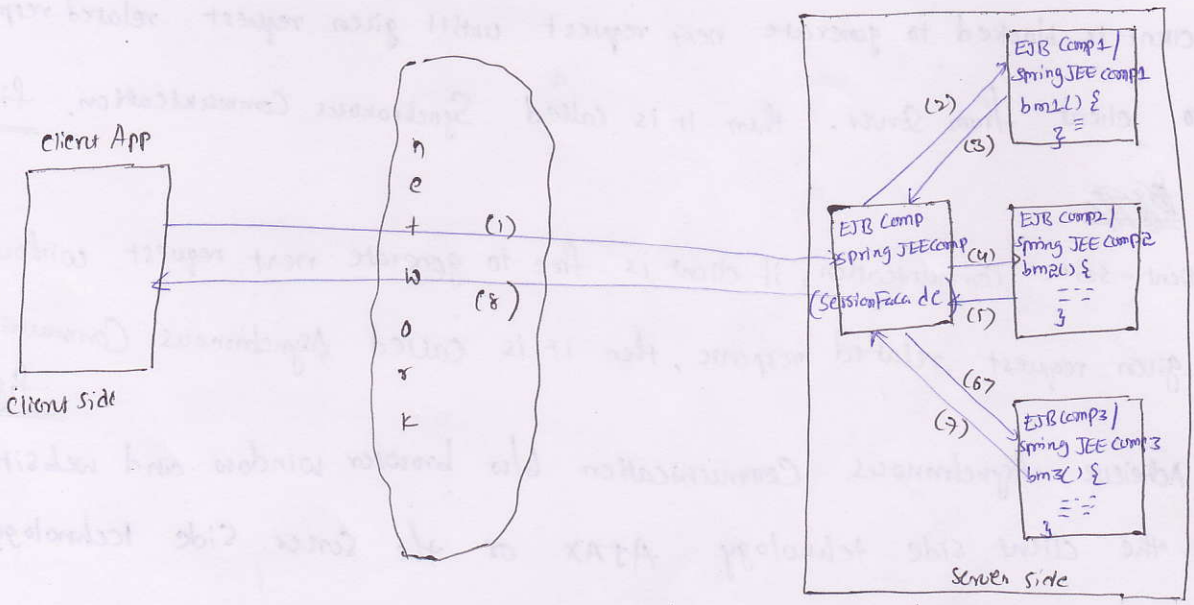
① Session Facade Design Pattern :-

(I)



3 network roundtrips are used here

(II)



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 ~~B~~ 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 "SessionFacade". SessionFacade 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 network round trip.

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

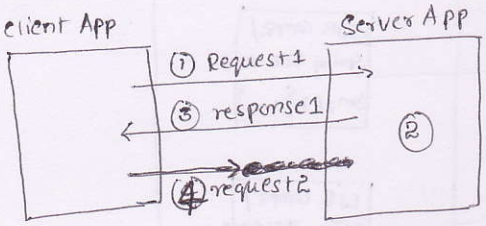


fig-1

Synchronous Communication b/w client Application and server Application

11/09/2012

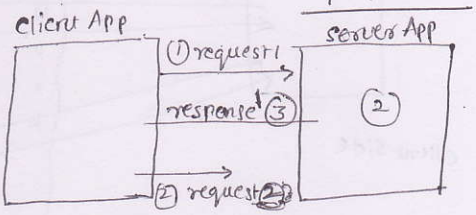


fig-2

Asynchronous Communication b/w client Application & server Application

→ If client is blocked to generate next request untill given request related response comes back to client from Server. then it is called Synchronous Communication. fig-I

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

* Message Facade :-

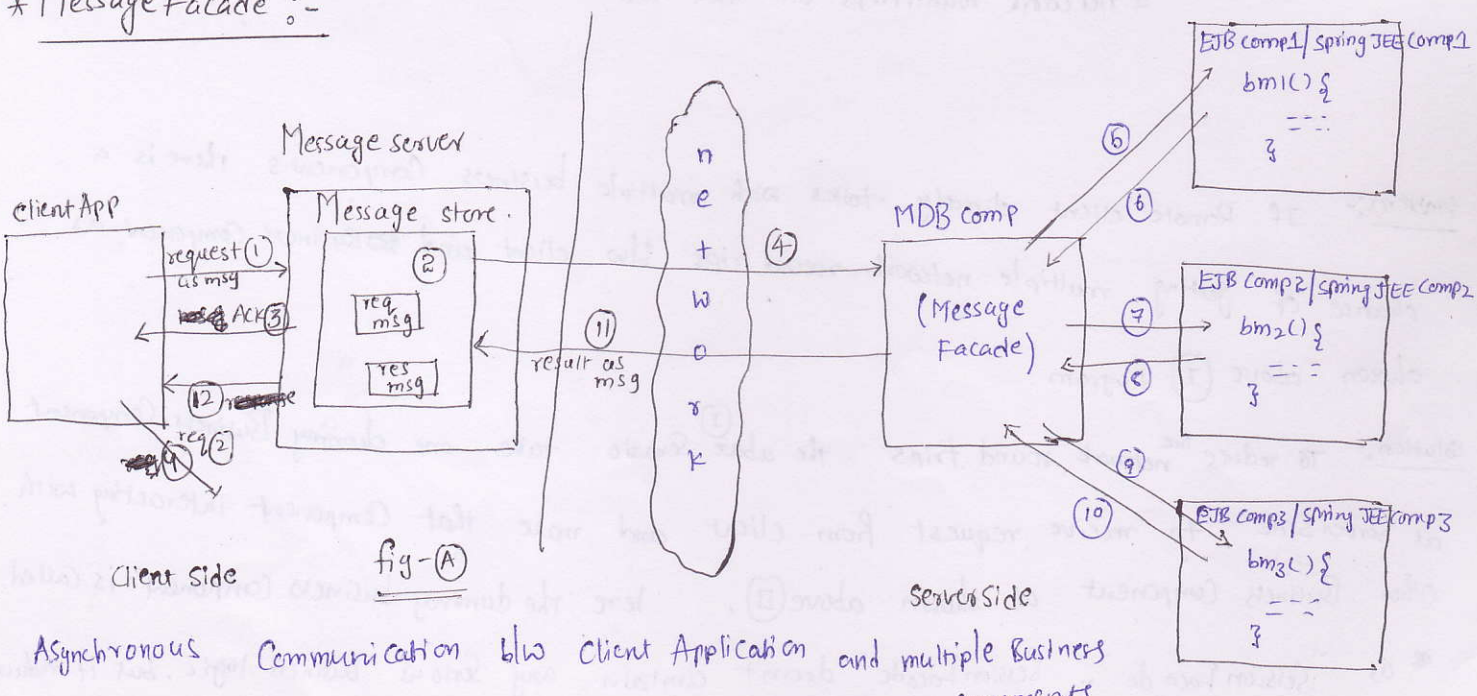


fig-(A)

Asynchronous Communication b/w Client Application and multiple Business Components

Problem:- The implementation of SessionFacade design pattern reduces network roundtrips b/w client and business Components, but that communication is Synchronous Communication. So, the client is has to wait to generate next request untill the result is received from multiple business Components.

Solution:- use JMS and MDB support to make Communication b/w client and bussiness Components more asynchronous as shown abave diagram fig-(A)

for related infomation on MessageFacade design pattern, refer page no:- 22,23

DAO: (Data Access Object):-

Problem:- Mixing up persistance logic with other logics of Application (Specially with busines 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 persistance logic from other logics of the application development and provides flexisility of modification.

DAO class contains the following logics.

1. Logic to establish the connection
2. Logic to release the Connection
3. Logic to perform CURD 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 persistance logics in DAO class we can use any persistance 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 persistance 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.
}  }
```

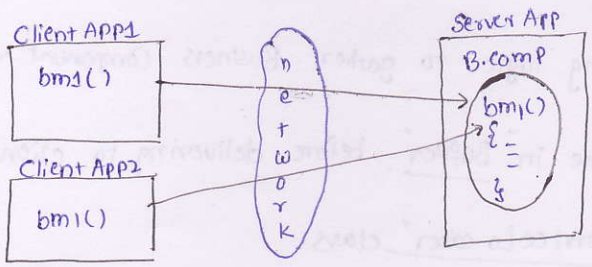
```
// DAO class for MySql
class MySqlDAO extends DAO
{
   public void makeConnection()
   {
     ----  logic to establish Connection with Mysql db s/w.
   }
   public void releaseConnection()
   {
     ---  logic to release Connection with Mysql db s/w
   }
  public int insertInfo(-,-,-,-)
   {
     ---  logic to insert record into Mysql db s/w
   }
  public int updateInfo(-,-,-,-)
   {
     ---
     ---  logic to update record in Mysql db s/w.
   }
}
```

→ 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.

12/09/2012

Bussiness Deligate :-

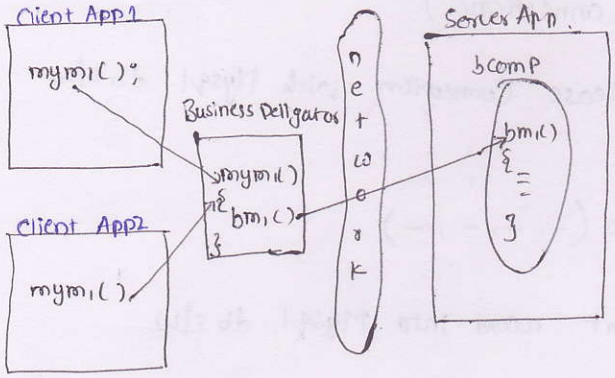
Problem :- Business In client applications calls business methods of business components available in Server application directly we don't get flexibility of modification If business method details are changed in future.



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

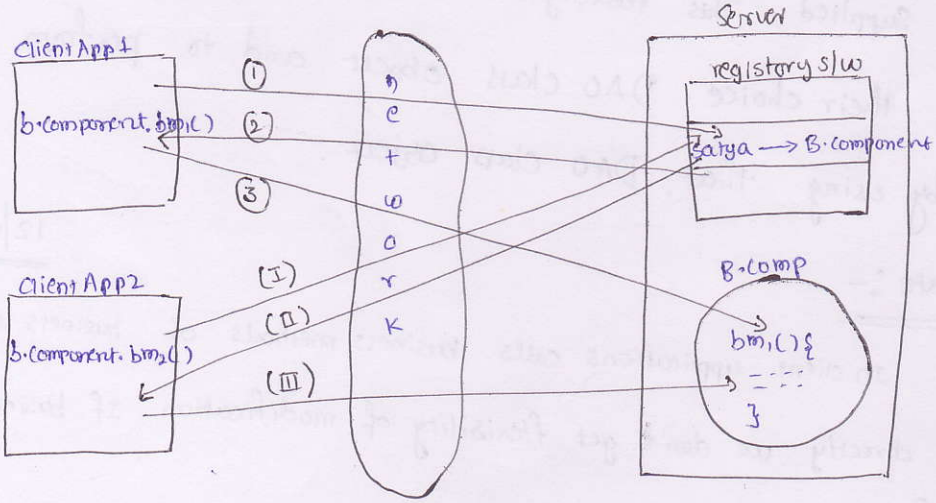
Business method can perform them Business Delegation class and there is no nessessarity of Disturbing client Applications.

→ In one angle it calls Struts ActionClass as Bussiness Deligate when it is having logic calling the bussiness 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 Bussiness Component references from registry s/w using JNDI code If Every client application Seperately gathers this Bussiness Compenent reference from registry that may increases more network round trips.



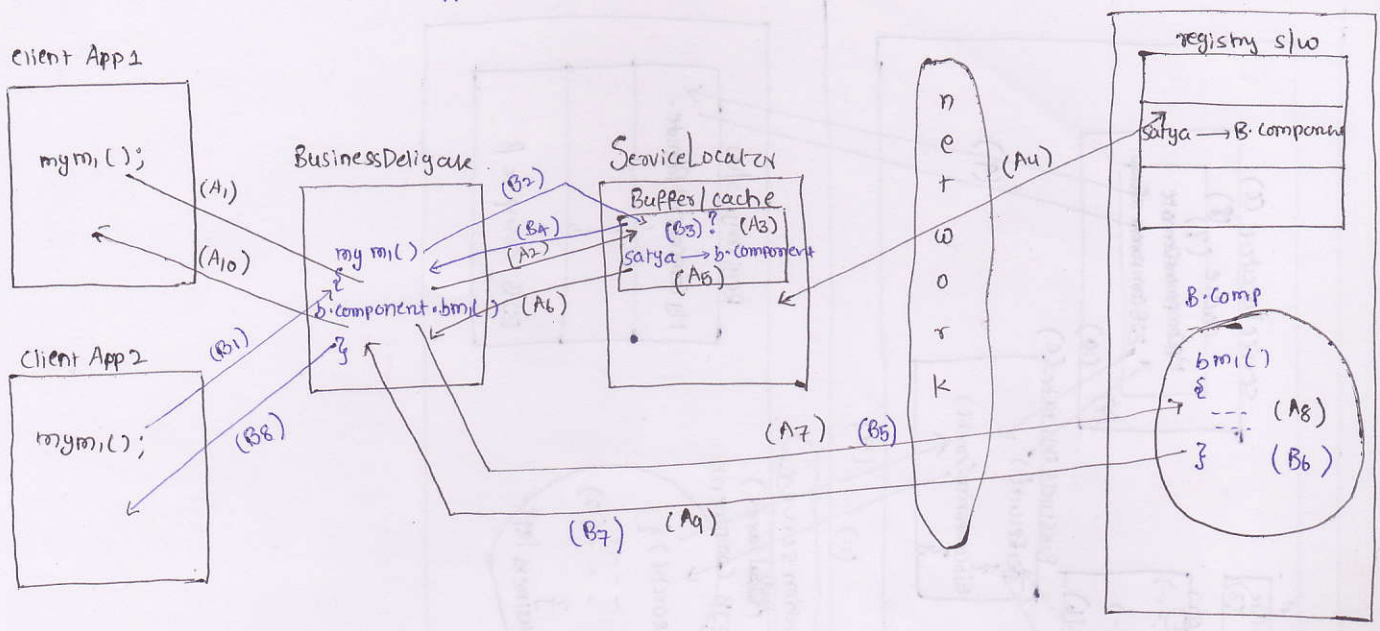
Solution :- Develop Helper class having logic to gather Business Component reference from registry of Server and to keep that one in "Buffer" before delivering to client Applications.

This helper class is called "Service Locater" class.

→ we generally develop this Service Locater class as Singleton java class to avoid multiple buffers to creations.

We generally used ServiceLocate class along with Business Deligate class.

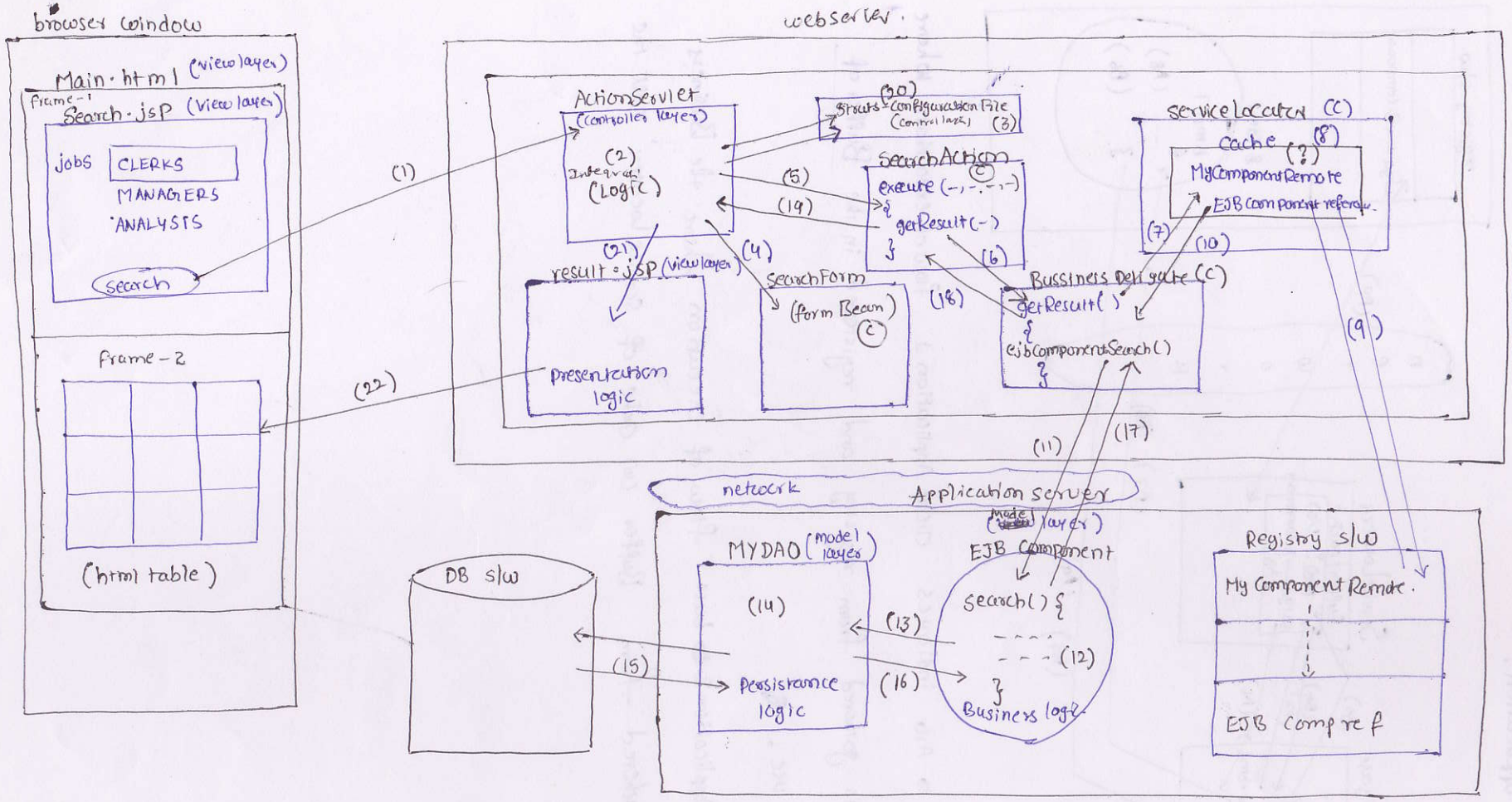
→ here Service Locater responsible to gather and give business object / component reference and Bussiness Deligate responsible to call Busines methods my using Busines object reference and to pass the results to client application.



In the above diagram A1 to A10 indicates client Application 1 flow of execution where Bussiness object reference is gathered from registry and registered in the Buffer of Service Locater before getting use.

B1 to B8 indicates client Application 2 base flow of execution where the Business Component reference is gathered from Buffer (or) cache of Service Locater call the Bussiness methods.

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



13/04/2012

W.r.to the diagram

(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 form data to FormBean class object.

(5) ActionServlet calls the execute() of ActionClass.

(6) This execute() passes the request to Bussiness Deligate class

(7)&(8) Bussiness Deligate uses ServiceLocater 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 s/w of Application Server and will be registered with the cache (or) Buffer of ServiceLocater.

(10) ServiceLocator passes the EJB Component reference to Bussiness Deligate class

(11) Bussiness Deligate class uses that EJB component reference and calls the Bussiness method of EJB Component.

(12)&(13)&(14)&(15)&(16) This Bussiness method takes the support of DAO class to send and execute SQL select query DataBase table and gets the records into ArrayList object.

(17) Bussiness method of EJB Component passes the result (ArrayList object) to Bussiness Deligate class

(18) Bussiness 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 2nd frame of webpage as HTML table content.

Explicitly implemented Design patterns of the above project :-

1. MVC2 — M — ejb component, V — jsp programs, C — ActionServlet
2. DAO
3. D.T.O class / V.O class
4. Bussiness Deligate
5. Service Locater
6. Singleton java class } required while implementing Servicelocater.
7. Factory method
8. Vieco 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. AbstractController (The pre-defined helper class to ActionServlet which is nothing but RequestProcessor class)
4. D.T.O class / V.O class (FromBean class)
5. Template Method (The process(-,-) of RequestProcessor)

For the above diagram base MiniProject having the implementation of DesignPattern refer the Suplementery HandOut given on 13/09/2012

Procedure to deployee and execute design project based mini project of bandaut

E:\DpProject
- web
  - StrutsApp
    - main.jsp
    - search.jsp
    - result.jsp
    - WEB-INF
      - lib
        - *.jar (10)
      - web.xml
      - s-c.xml
      - classes
        - SearchForm.class
        - SearchAction.class
        - myfile.properties
        - ServiceLocator.class
        - BussinessDeligate.class
        - *.java (4)
        - comp
          - MyCompRemote.class (Copy from EJBcomponent)
        - P1
          - EmpBean.class (Copy from EJB component)

Jarfiles in classPath :-

Servlet-api.jar

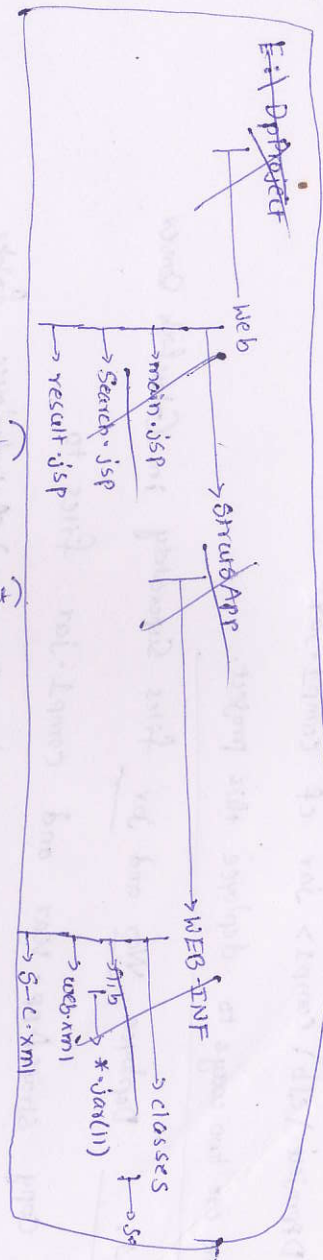
Struts-core-1.3.8.jar

Jarfiles in WEB-INF\lib folder

10 reguler Struts jar files

Prepare war file representing struts Application

e:\DPproject\web\strutsApp>jar cf strutsApp.war



E:\DPproject
- ejb
  - comp1
    - META-INF
    - *.java (4)
    - comp
      - MyCompRemote.ja
      - myCompBean.
      - myDAO.class
    - P1
      - EmpBean.class

Jarfiles in <Glassfish-home>\Appserver\domains\mydomain1\lib\ext

jar files in class path

javaee.jar

ojdbc14.jar

Prepare jar files representing EJB Component :-

E:\DPproject\ejb\comp1> jar cf comp1.jar

there are two ways to deployee this project.

Approach 1 :- Deployee War and Jar files Seperately in Glassfish Server

Copy StrutsApp.war and comp1.jar files to

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

Request URL: http://localhost:8080/mywebapp

Open browser window

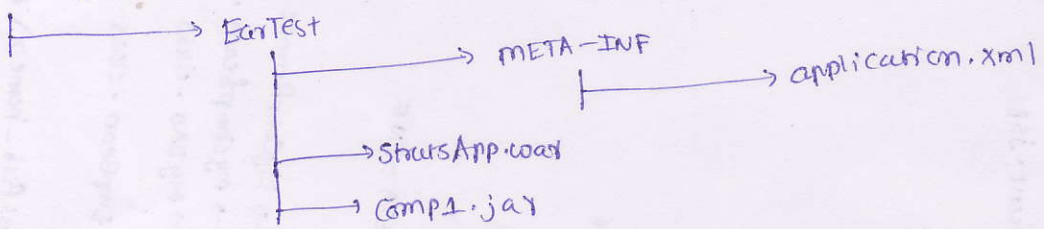
Approach 2 :- Combine War and Jar files Ear file and deployeeing Glassfish Server.

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 :- (DD file for Ear file)

<!DOCTYPE application PUBLIC

"-//EN" "http://java.sun.com/dtd/application_1_3.dtd">

<application>

<display-name> earTest </d-n>

<description> Struts EJB project </description>

<module>

<ejb> comp1.jar </ejb>

</module>

Mini Project implementing Design pattern 13/09/2012 (1)

```
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">  -> *1 (ref 7)
13                  (5) -> Action Path of Action class (refer line no: 69)
14    <bean:message key="my.lbl"/>
15                  -> refer: 31
16    <html:select property="job" multiple="yes">
17       <html:option value="CLERK">CLERKS</html:option>
18       <html:option value="MANAGER">MANAGERS</html:option>
19       <html:option value="ANALYST">ANALYSTS</html:option>
20       <html:option value="SALESMAN">SALESMEN</html:option>
21    </html:select>
22
23       <html:submit>
24          <bean:message key="btn.cap"/>
25       </html:submit>
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---------- Deployement Descriptor File
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-cofig.xml---------- Struts Configuration file (Controller)
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

(4)

*1) when form is submitted to web Application the output Generated by the web resource program of web App will be displayed in frame whose name is "f2"

38-51 (6)

Form bean configuration

69 to 74 Struts Action class cfg (7) -> (11)

```
        <forward name="success" path="/result.jsp"/>
      </action>
    </action-mappings>

    <message-resources parameter="myfile"/>

  </struts-config>
```

SearchForm.java — FormBean (Controller layer)

```
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionForm;

public class SearchForm extends ActionForm {
   private String job[];

   public void reset(ActionMapping mapping,HttpServletRequest req)
   {
      job=new String[1];
      job[0]="no item selected";
   }

   public String[] getJob() {
      return job;
   }

   public void setJob(String[] job) {
      this.job = job;
   }

}
```

SearchAction.java — Action Class (Controller layer)

```
import java.util.ArrayList;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public class SearchAction extends Action {
   @Override
   public ActionForward execute(ActionMapping mapping, ActionForm form,
         HttpServletRequest request, HttpServletResponse response)
         throws Exception {
      //read form data
      SearchForm sf=(SearchForm)form;
      String jobs[]=sf.getJob();
      System.out.println("In execute(-,-,-,-):SearchAction");
      System.out.println("before calling B.method");

      // call B.method
      BusinessDelegate bd=new BusinessDelegate();
      ArrayList al=bd.getResult(jobs);
      System.out.println("after calling B.method");

      // send result to result page as req attribute
      request.setAttribute("result",al);
      //forward control to result page
      return mapping.findForward("success");
   }//execute(-,-,-,-)
}//class
```

ServiceLocator.java — class implementing ServiceLocator Designed Pattern (Control layer)

```
import java.util.*;
import javax.naming.*;

public class ServiceLocator
{
  private Hashtable Cache;
  private InitialContext ic;

  private static ServiceLocator sl=null;
```

```
139
140   private ServiceLocator() //private constructor (to support Singleton)
141   {
142    try
143    {
144          Cache = new Hashtable();
145
146      // jndi properties
147         Hashtable ht=new Hashtable();
148           ht.put(Context.INITIAL_CONTEXT_FACTORY,
149                        "com.sun.enterprise.naming.SerialInitContextFactory");
150         ht.put(Context.PROVIDER_URL,"iiop://localhost:4848");
151           ic=new InitialContext(ht);
152    }//try
153    catch(Exception e)
154    {
155     e.printStackTrace();
156    }
157   }//constructor
158
159  // Factory methid 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()
167
168   // method having ServiceLocator Impl logic
169   public Object getService(String jndiName)throws Exception
170   {
171    System.out.println(" ServoceLocator:getService(-) ");
172    try
173    {
174      if(!Cache.containsKey(jndiName))
175      {
176       Cache.put(jndiName,ic.lookup(jndiName));
177      }//if
178    }//try
179    catch(Exception e)
180    {
181     e.printStackTrace();
182    }//catch
183
184     // Return object from cache
185    return Cache.get(jndiName);
186
187  }//getService()
188 }//class
189 ----------BusinessDelegate.java----------
190 import java.util.*;
191 import comp.MyCompRemote;
192
193 public class BusinessDelegate
194 {
195
196      public  ArrayList getResult(String job[])throws Exception
197      {
198           System.out.println("getResult method of BusinessDeligate class");
199
200        ServiceLocator sl= ServiceLocator.getLocator();
201        Object obj  =sl.getService("comp.MyCompRemote");
202        MyCompRemote wor=(MyCompRemote)obj;
203       // Calling B.method
204        ArrayList al=wor.search(job);
205      return al;
206      }
207 }
```

(16) logic to Establish connection with Glass Fish registry by using its JNDI properties

(17) (18)

(14) (15) (24)

ServiceLocator contains one member variable "cache"
Cache contains Buffer

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.

from Hash table registry

gathers EJB component reference from registry.

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

class implementing Bussiness Deligate DesignPattern (Control layer)

Use the single ton object of ServiceLocator

logic to get EJB component reference from the cache maintained by ServiceLocator

Calls the bussiness method of EJB component

returns to execute method of Actionclass.

result.jsp (27)

Result Page

```
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 weather 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 'id' 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---------- Bussiness Interface of EJBComponent (Model layer)
5  package comp;
6  import java.util.ArrayList;
7  import javax.ejb.Remote;
8                          └ annotation
9  @Remote → makes Bussiness interface as Remote Bussiness 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          └ D.T.O. class Visibility.
19 @Stateless → makes the Bean as SLST (Stateless session Bean)
20 public class MyCompBean implements MyCompRemote {
21                          (20) → Bussiness method
(19) 22    public ArrayList search(String[] jobs) {
23       System.out.println("MyCompBean:serach(-) 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')
37
38            // use the persistance logic of DAO clas
refer line nos 54 to 93
39       MyDAO dao=new MyDAO();  ——refere line no's 54 to 93
(21) 40  ArrayList al=dao.findEmployees(cond);
41        return al;
42    }//search  └ returns to BusinessDelegate class.
                (23)
43
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    {                    └ method having persistance logic
56
57     System.out.println("findEmployees(-):MyDAO");
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 "
```

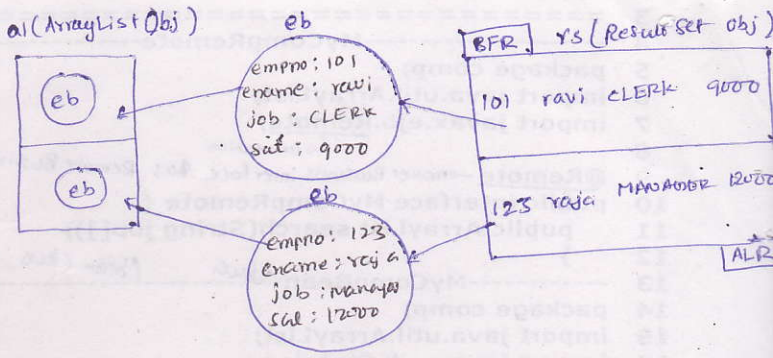
54-93 (22)

job[]
CLERK 0
MANAGER 1

Select empno, ename, job, sal from emp where job in ('CLERK','MANAGER') order by job

```
                    + cond+" order by job");
        while(rs.next())
         {
         System.out.println("In while");
          EmpBean eb=new EmpBean();

            eb.setEmpno(rs.getInt(1));
            eb.setEname(rs.getString(2));
            eb.setJob(rs.getString(3));
            eb.setSal(rs.getFloat(4));

          al.add(eb);
         }//while
            rs.close();
         st.close();
            con.close();
        }//try
        catch(Exception ee)
         {
         System.out.println(ee.toString());
         }

        return al;
      }//findEmployees
   }//class
```

logic to transfer the records of ResultSet Object to Arraylist by taking the help of D.T.O class



returns to the search method of EJB Component Bean class.

D.T.O class | V.O. class.

```
-----------------------------------EmpBean-----------------------------------
package p1;
import java.io.*;

public class EmpBean  implements Serializable
{
```

mandatory

```
      private int  empno;
      private String ename;
      private String  job;
      private float sal;

   public int getEmpno() {
      return empno;
   }

   public void setEmpno(int empno) {
      this.empno = empno;
   }

   public String getEname() {
      return ename;
   }

   public void setEname(String ename) {
      this.ename = ename;
   }

   public String getJob() {
      return job;
   }

   public void setJob(String job) {
      this.job = job;
   }

   public float getSal() {
      return sal;
   }
   public void setSal(float sal) {
      this.sal = sal;
   }
}
```

When we deployee EJB 3.x component in GlassFish Server when the Bussiness 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 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

A Perfect Place for All Java Resources