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

Struts

(Natraj Notes)

www.tutorial4us.com

01/11/2012

Understanding the basics of web application

Q:- What is the diff b/w webapplication and website ?

Ans:- Webapplication that is hosted (→ placed) on the internet network by having domain name. (WWW.XYZ.com). is called as website.

→ a webapplication is a collection of webresource programs.

These webresource programs generate webpages. Based on the type of the webpage they generate there are two types of webresource programs.

① Static Webresource prgs :-

generate static webpages

Eg:- html prgs.

② Dynamic webresource prgs :-

generate Dynamic webpages

Eg:- servlet prgs, jsp prgs, php prgs, asp.net prgs and etc...

images files, text files are helper webresource prgs to other web resource prgs because they can't generate webpages directly.

→ Website contains both static and dynamic webpages

→ so, in the development of web application both static and dynamic web resource prgs will be used.

→ To Develop and execute web application we need following s/ws

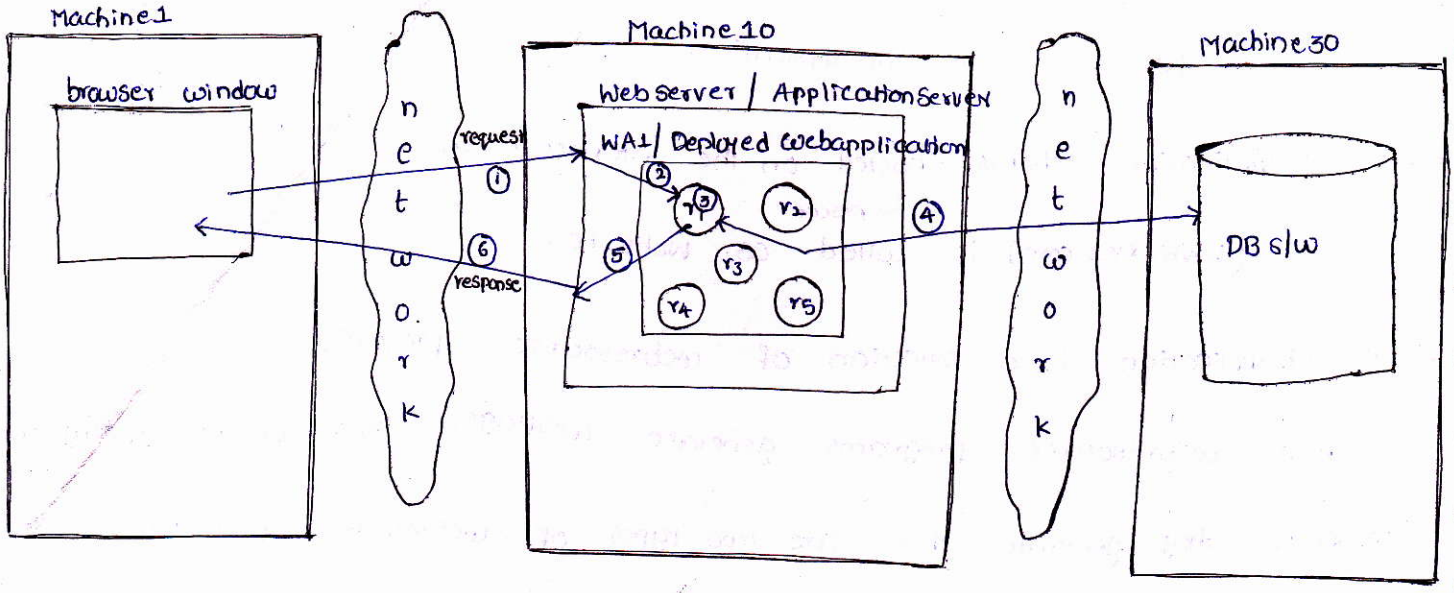
① Browser s/w (to get Browser windows as web clients)

② Web Technologies (to develop webresource programs)

③ Web Server / Application Server s/w (to manage and execute deployeed web app's).

④ DataBase s/w (Optional)

Understanding the flow of execution in webapplication :-



r_1, r_2, r_3, r_4, r_5 are the web resource programs of webapplication.

*) Based on the place where webresource programs execute when it is requested by client (Browser window). There are 2 types of webresource programs.

① Client Side webresource prgs

② Server Side webresource prgs (execute in the server when requested)

Eg:- Servlet prgs, Jsp prgs, php prgs and etc...

Client Side web resource prgs :- Come to browser window from webserver for execution when requested.

Eg:- html prg, java script prgs and etc...

NOTE:- generally all static webresource prgs are Client side webresource prgs. Simillerly all Dynamic webresource prgs are Server side webresource prgs.

Different Approches of Developing java Web application

1. Model 1

→ only jsp prgs or only servlet prgs will be there as Server side web resource prgs of web application.

→ If Servlet prgs are placed in Webapplication then Jsp programs should not be placed in webapplication

2. Model 2 (MVC)

→ Here we must use multiple technologies support to develope the logics in multiple layers of webapplication.

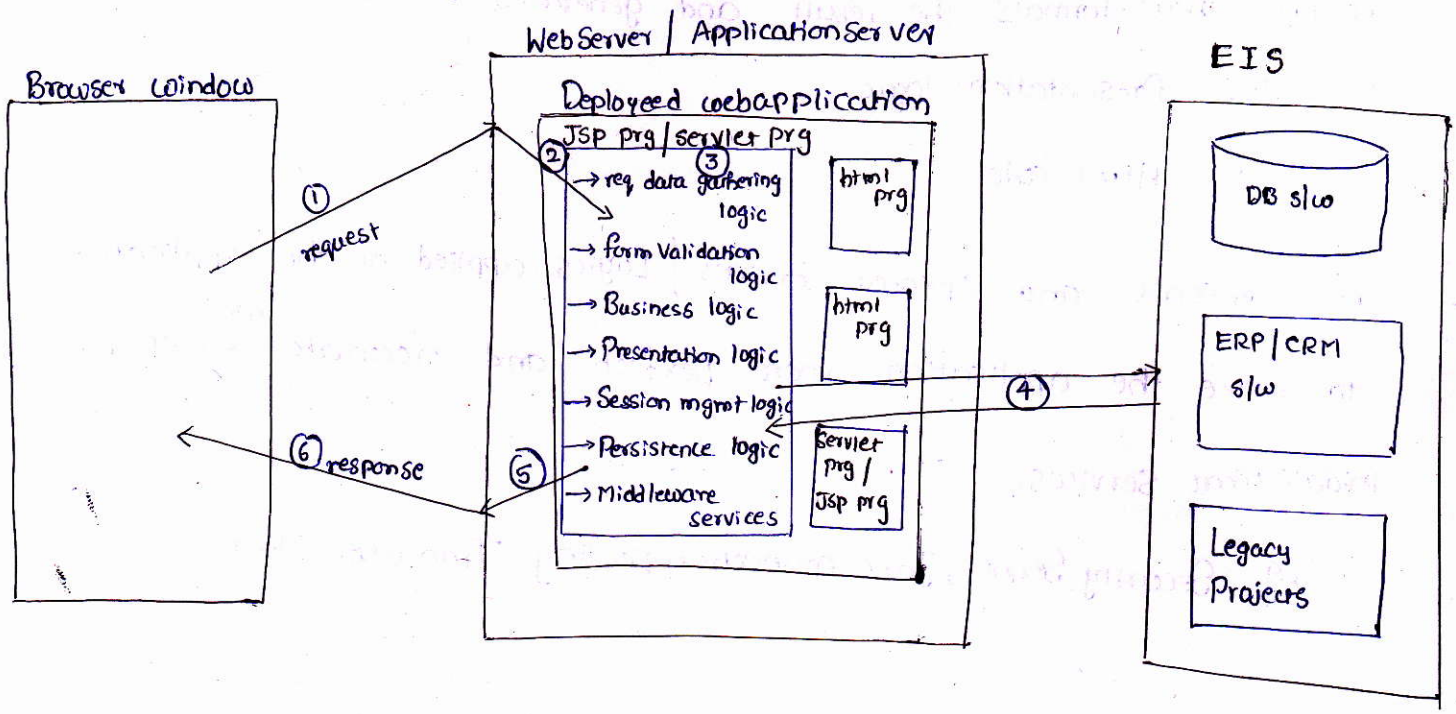
The Layers are Model layer (M)

and vice versa

→ Software industry prefers using jsp prgs as server side web resource prgs while developing Model1 approach/ Architecture based web application.

View Layer (V), Controller Layer (C).

Developing Model1 Architecture Based Web application :-



→ The logic that reads various details from client generated request is called as RequestDataGrathering Logic. These details are Request Header Values, FormData and etc....

→ The logic that validates pattern and Format of the form Data is called as FormValidation Logic.

Eg: checking weather EmailId is having '@', '.' symbols or not.

→ The Main Logic of the application is called as Business Logic (or) Request processing Logic.

→ The Logic that interacts with DataBase s/w to manipulate the data is called persistence logic.

Eg: Jdbc code, Hibernate code and etc...

→ The logic that remembers the client data across the multiple request during a session is called SessionManagement Logic.

Eg:- The logics of all SessionTracking techniques

- → HiddenFormFields
- → Cookies
- → HttpSession and etc...

→ The logic that formats the result and generates the user interface for End user is called Presentation logic.

Eg:- Html code

→ The additional and optional Services | Logics applied on the application to make the application more perfect and accurate are called as middle ware Services.

Eg:- Security Service, Jdbc connection pooling, TransactionManagement and etc..

→ One project can take the support of multiple backends. All these Backends together is called as EIS (Enterprise Information System).

→ The project that is developed by using old Technologies (or) By using the old versions of existing technologies is called as legacy project.

Eg:- Cobal project, Java 1.2 project and etc...

→ SAP is ERP slw. (Enterprise Resource plan)
Siebel is CRM (Customer Relation Management) slw.

→ In Model1 Architecture based webapplication Developement every Server Side web resource contains multiple logics. due this the following limitations and advantages are there.

02/11/2012

Limitations :-

→ Multiple logics will be Mixed up in every Server Side webresource program. This indicates there is no clean Separation blw Logics.

→ Modifications done in one kind of logics may effect other kind of logics.

→ Maintanance (Bug fixing) and enchancement of project becomes Complex process.

→ Parllel development is not possible. so, the productivity is very poor.

NOTE :- Doing more work in less time is called good productivity

→ Certain middle ware Services must be implemented by the programer manually. This increases burden on the programer.

Advantages :-

→ knowledge on either Servlet or Jsp is enough to webapplication.

→ Since parllel development is not possible multiple programers are not required to develop the web application.

Model1 Architecture is suitable to develop small scale web applications (max of 10 pages)

Model2 (MVC) Architecture :-

M → Model Layer → (Contains B.logic and persistence logic)
→ (use java class / java bean, ejb, spring, spring with hibernate and etc... to develop these logics)
→ (Model Layer is like Account Officer)

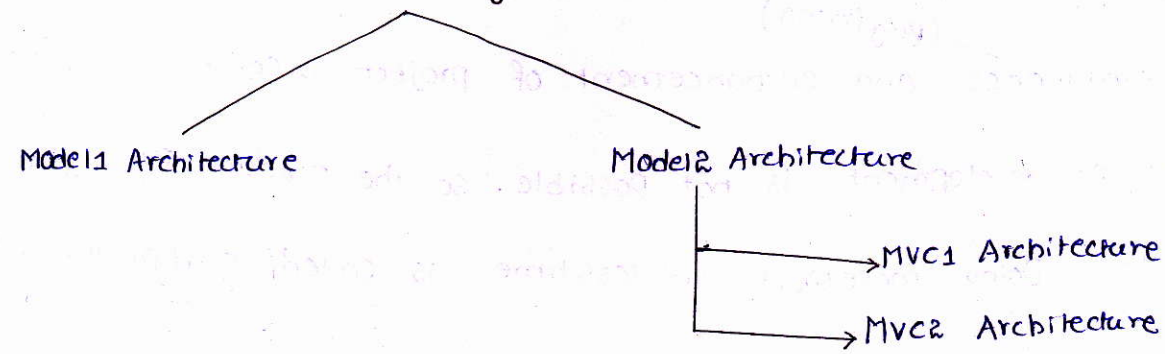
V → View Layer → (Contains Presentation logic)
→ (use html, jsp, velocity, freemarker and etc... to develop this logic)
→ (View Layer is Beautician)

C → Controller Layer → (Contains Integration logic)
→ (use Servlets / Servlet Filter to develop this logic)
→ (Controller Layer is like Traffic police)

Integration logic is the central logic of the webapplication which takes the

request from client, passes them to model Layer resources, gather the results from Model Layer resources and passes the results to View Layer resources. This logic Controles and monitors Every activity in a Web application execution.

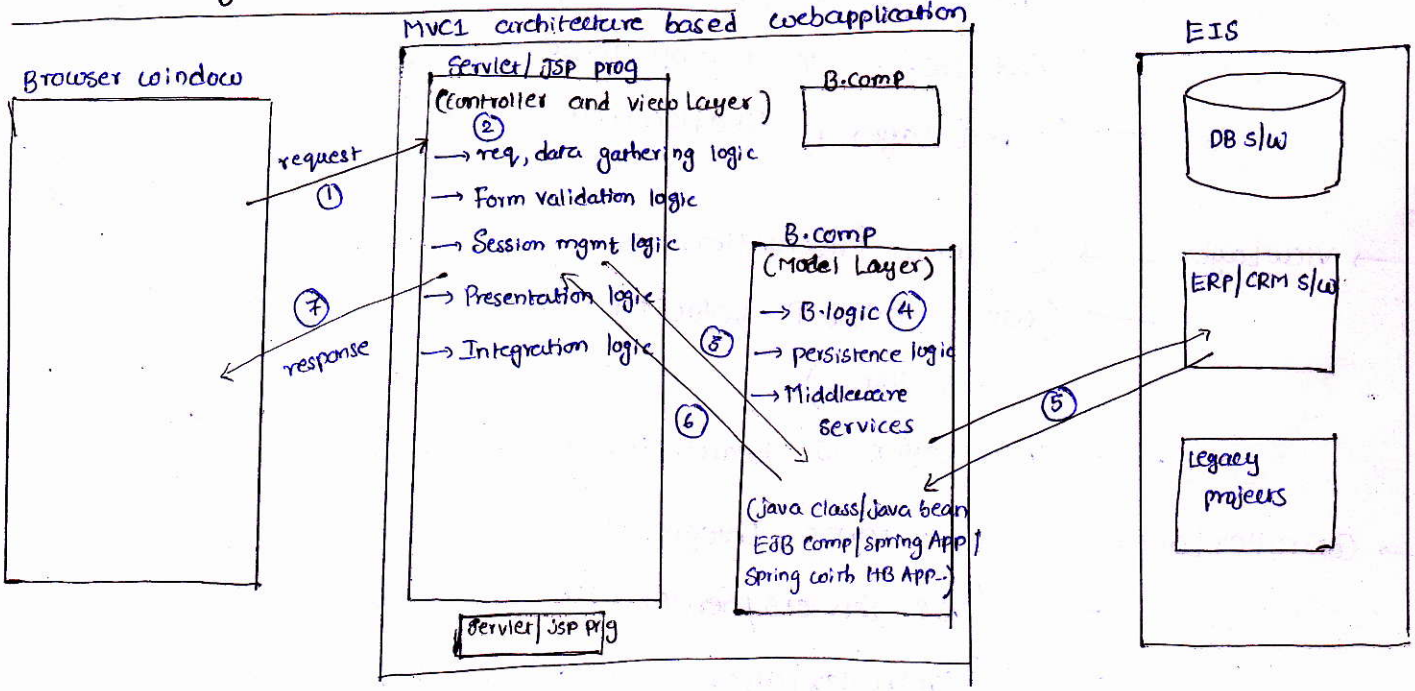
Different Models of Developing Java webapplications



→ In MVC1 Architecture
- → we take Separate Model Layer resources having b.logic and persistence logic.
- → we take Single resources as view and Controller layer resource having presentation logic and Integration logic.

→ In MVC2 Architecture
- → we take Separate Model Layer resources (b.logic and persistence Logic).
- → we take Separate view layer resources (presentation Logic)
- → we take Separate Controller layer resources (Integration Logic)

Understanding The MVC1 architecture :-



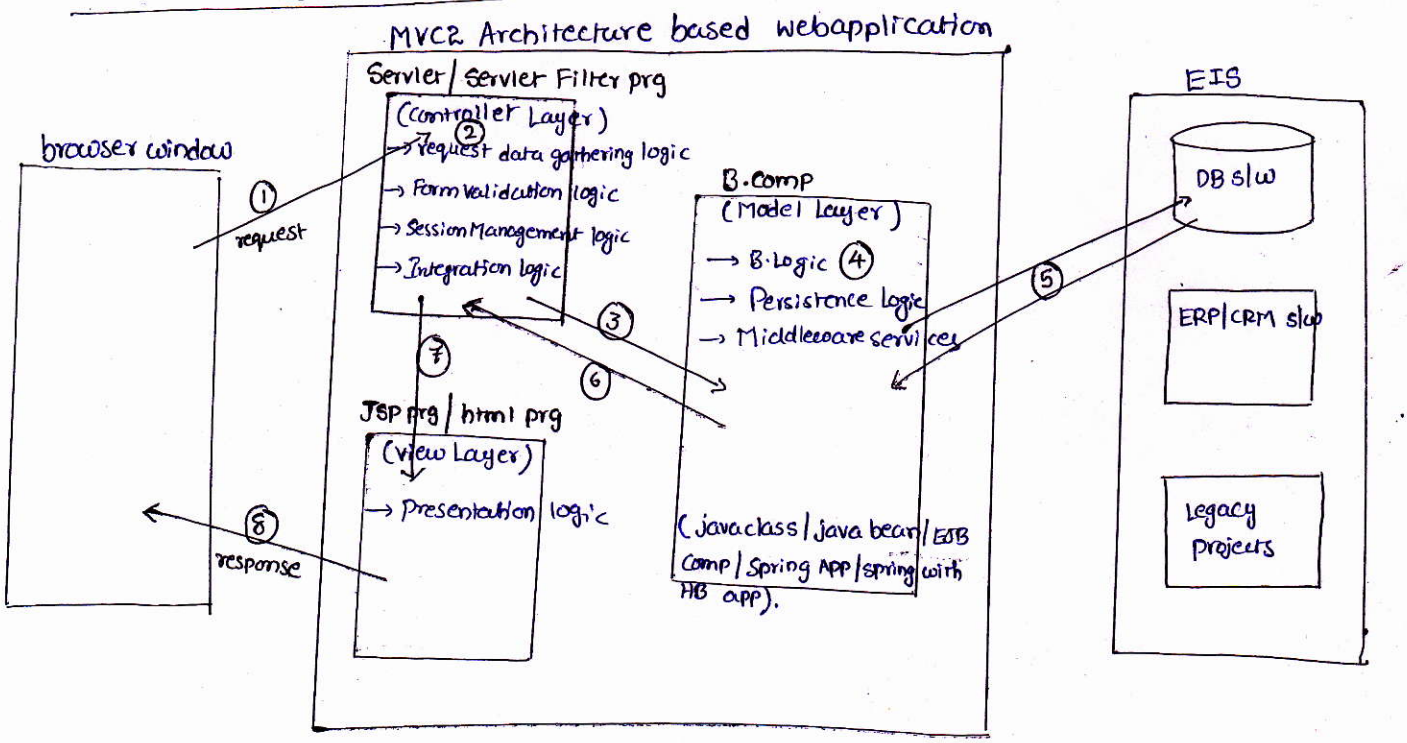
→ MVC1 Architecture gives clean separation of logics compared to Model1 Architecture. Because we can write B.logic and persistence logic separately in ModelLayer B.components while working with MVC1 Architecture.

To get More clean Separation of logics it is recomended to use MVC2 architecture to develop the webapplications.

NOTE:- MVC1 Architecture is good to develop medium Scale websites (Max of 30 pages)

Eg:- WWW.roseindia.net
WWW.w3schools.com

→ UnderStanding MVC2 Architecture :-



06/11/2012

→ What is Framework ?

→ "Framework is a special slw. i.e. develop based on core technologies having the ability to generate the common logics of application dynamically based on other application specific logics given by programmer."

→ "Framework slw provides abstraction layer on core technologies and simplifies the process of application development for programmers. Every framework slw internally uses certain core technologies but it makes programmer not to worry about core technologies (nothing but abstraction Layer) while developing flw based slw application."

If we develop mvc2 architecture based web application by taking Servlet, Jsp core technologies then all the logics of all the layers must be developed by the programmers manually from Scratch level. If you develop same application by using Struts flw then the controller layer indication logic will be generated dynamically and programers just needs to Consentrate only on View layer, model layer logics development.

This improves the productivity of web application development.

→ 3 types of flw's in Java environment :-

① Web flw's :-

→ provides abstraction layer on Servlets, Jsp technologies.

→ develops mvc2 architecture based web applications.

→ Eg:- Struts → from apache slw foundation (1)
JSF (JavaServer Faces) → from Sun micro system (2)
WebWork → from Opensymphony
Spring mvc → from Interface21 (2)
Tapstrey → from Adobe
ADF → from Oracle corp (Application Development Frame)

2) ORM F/w's :-

→ Provides abstraction layer on Jdbc Technology.

→ useful to develop objects based DB s/w independent persistence logic (Object Relation mapping (ORM) persistence logic)

→ Eg:- Hibernate ⟶ from SoftTree (Red Hat)

Toplink ⟶ from Oracle Corp

iBatis ⟶ from apache foundation

JPA (Java Persistence Api) OpenJPA ⟶ from Sun microsystem.

OJB (Object Java Beans) ⟶ from apache s/w foundation

(Java Data Objects) JDO ⟶ Java Adobe

⋮

3) java-jee f/w :-

→ provides abstraction layer on multiple Java, JEE technologies like jdbc, servlets, jsp, ejb, rmi, jms and etc...

→ useful to develop all kinds of Java, JEE applications.

Eg:- spring ⟶ from interface 21

note:- .NET is also a f/w.

Struts :-

type : java based web f/w s/w to develope mvc2 architecture based webapplications

Version : 1.3.x (Compatible with jdk 1.4+)

2.x (Compatible with jdk 1.5+)

Vendor : apache foundation

Creator : mr. craig

Open Source s/w: download s/w as Zip file from www.apache.org website.

Zip file name : Struts-1.3.8-all.zip

Struts-2.3.4-all.zip

Installation of s/w: Extract the Zip file to a folder.

reference books : Jakartha Struts (1.x) → from O'reilly press

Black book of Struts 2 ⟶ from Tata MGr

Definition of Struts :-

Struts is a Java based web flw slw. i.e. providing abstraction layer on
↳ Hiding the implementations
the core servlet, JSP technologies to develop MVC2 architecture based Java web applications having the capability to generate the Integration logic of controller layer dynamically based on the programer supplied logics of view, model layers.

07/11/2012

Struts 1.x slw installation gives

→ Base Flw

→ Additional plugins
(validator plugin and Tiles plugin)

→ plugin is a Batch slw or slw application that can enhance the functionalities of existing slw or slw applications. In Java env plugins come as jar files

→ Example applications

→ Documentation

→ Jsp Tag libraries (5 no. of jsp tag Libraries)
Jsp Tag library contains set of readyly available jsp tags

→ Source code

- <Struts 1.x_home>\apps ⟶ gives example Struts Applications
- <Struts 1.x_home>\docs ⟶ gives api docs, JSP Taglibrary docs, DTD
- <Struts 1.x_home>\lib ⟶ jar files representing Struts libraries
- <Struts 1.x_home>\lib\ struts-core-1.3.8 jar file representing Struts 1.x api and this jar files is having multiple dependent jar files

→ <Struts 1.x_home>\src ⟶ gives source code of Struts slw.

If Java web application uses 3rd party API in its webresource programs then the 3rd party API related main jar files should be added to CLASSPATH and the 3rd party API main and dependent jar files must be added to WEB-INF|Lib folder of web application. here jar files added to CLASSPATH will be used by Java Compiler

To recognize 3rd party API during the compilation of web resource program.

Similerly jar files added to WEB-INF\lib folder will be used by Servlet Container ~~and~~ Be and JSP Container to recognize and use 3rd party API during the execution of web-resource programs.

└ other than JdK API's

Example Scenario :-

First.jar (Third party API given by vender 1)

Test.java

```
public class Test
{
    public void m1()
    {
        Demo d=new Demo();
        d.m2();
    }
}
```

Second.jar

Demo.java

```
public class Demo
{
    public void m2()
    {
        ------
        ------
    }
}
```

Second.jar file is dependent jar file to First.jar because the "Test" class of First.jar is using the "Demo" class of second.jar

Java webapplication

FirstSrv.java

```
public class FirstSrv extends HttpServlet
{
    public void service(-,-) throws SE,IOE
    {
        ------
        ------
```

```
Test t = new Test();
t.m1();
--------
--------
}
}
```

>javac FirstSrv.java (X)

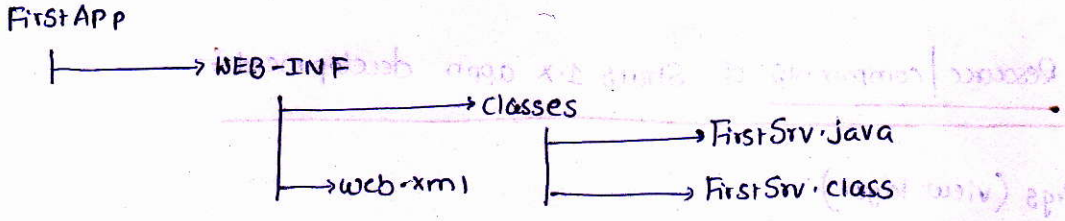
Can not find symbols HttpServlet, ServletException, ... and Test, m1

add servlet-api.jar and First.jar files to classpath

>javac FirstSrv.java (Success)

gives FirstSrv.class

Deployeement Directory Structure :-



Deployee FirstApp webapplication in server

give request to FirstSrv prg :

http://localhost:8080/FirstApp/Test1 (X)

↳ url pattern

java.lang.NoClassDefFoundException: Test

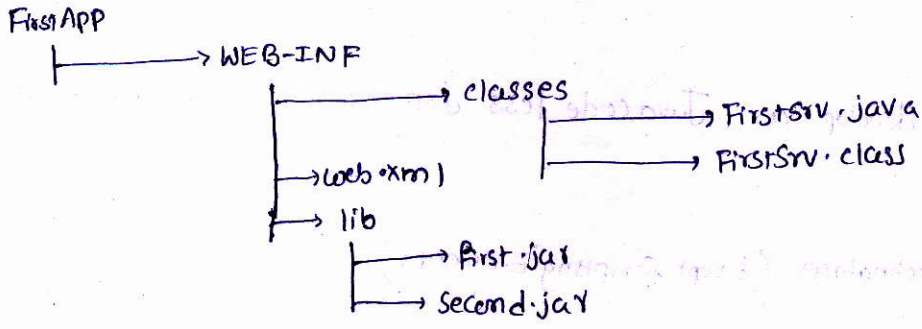
add First.jar file to WEB-INF\lib folder of FirstApp webapplication

give request to FirstSrv Prg :

http://localhost:8080/FirstApp/Test1 (X)

java.lang.NoClassDefFoundException : Demo

also added second.jar to WEB-INF\lib folder of FirstApp webapplication.



give request to FirstSrv prg:

http://localhost:8080/FirstApp/test1 (Success)

→ Developing Struts application is nothing but Developing Normal web application having Struts API based web resource prgs. for this we need to the Struts api related main jar file (Struts-core-1.3.8.jar) to classpath and the Struts api related main and dependent jar files to WEB-INF\lib Folder

08/11/2012

→ Using Struts F/w we can't develop stand alone applications, Desktop applications, two-tier applications. it can be used only develop MVC2 architecture based Java web applications

→ The important Resource/components of Struts 1.x appn development :-

1) JSP Comps/Prgs (view layer)
2) ActionServlet (Controller layer)
3) web.xml (Deployement Descripter (DD) file)
4) FormBean class/ActionForm class (Controller layer)
5) Action Class (Controller Layer/Model Layer)
6) ActionForwards (Controller Layer)
7) Struts-cfg file (Controller Layer)

① JSP Comps/Prgs (view layer):-

→ represents the resources of view Layer having presentation logic

→ presentation logic is required to prepare user interface for End user. Using this UserInterface the End user can supply inputs to Struts application and can view results given by Struts application.

→ It is recomended to develop as Java code less jsp programs by using the following jsp tags

→ Built-in JSP tags jsp technology (Except Scripting Elements)

→ JSTL tags

→ Struts supplied Jsp tags

→ Third party supplied Jsp tags

→ Custom Jsp tags

note:- We can also use html, freemarker, velocity technologies to develop the view layer resources.

2) ActionServlet (Controller layer) :-

→ It is built-in Servlet Controller of Struts based web appn.

Org.apache.struts.action.ActionServlet
↳ pkg name

→ It traps all the request coming to struts application passes them to appropriate Model layer resources, gathers results from ModelLayer resources and passes the results to appropriate View layer resources.

→ The Integration logic of this Controller Servlet will be generated dynamically based on the Configurations done in Struts Configuration file.

→ Even though it is built-in Servlet its Configuration in web.xml is mandatory.

3) web.xml (DD file) :-

→ It is Deployement Descriptor file (or) web application configure file.

→ This file contains ActionServlet Configuration (mandatory) welcome file configuration (optional) Jsp library configuration (optional) and etc.

NOTE:- The process of specifying resource details to make the underlaying container (or) Server (or) Flw (or) run-time environment recognizing the resource is called as resource configuration.

4) FormBean class / ActionForm class (Controller layer) :-

→ It is a programmer supplied Java Bean class extending from org.apache.struts.action.ActionForm (AC)

→ It is a Javabean having the capability to receive and hold the form data given by form page through Action Servlet.

→ ActionServlet creates this FormBean class obj and copies Form data after receiving Form data from form page.

→ FormBean class is optional in Struts application but recomended to use.

→ This FormBean class contains logic to hold form data, logic to reset the form data and the logic to validate the form data.

Q:- What is the need of FormBean class in Struts application?

According to MVC2 Controller Servlet is responsible to have FormValidation logic. In Struts applications the Controller Servlet is pre-defined ActionServlet class. so, we can't place FormValidation logic in ActionServlet.

To overcome this problem ActionServlet copies the received form data to the programer supplied FormBean class. so, programer can validate form data in FormBean class

5) Struts Action class (Controller Layer | Model Layer) :-

→ It is the java class that extends from org.apache.struts.action.Action (c)

→ when B.logic and persistence logics are directly placed in Action class then it acts as Model Layer resource.

→ If Action class contains logics to interact with Model Layer components like EJB components, Spring Applications, Spring with Hibernate applications and etc... then Action class is called as Controller Layer resource.

→ Every request trapped by ActionServlet will goto one (or) other Struts Action class.

Q:- What is need of Struts Action class?

According to MVC2 the Controller Servlet should contain integration logic to communicate with Model Layer resources. In Struts applications Controller Servlet is a predefined ActionServlet class. so, programers can't place Integration logic in ActionServlet.

To overcome this problem ActionServlet passes the treaped request to Struts Action class. so, that programer can use Struts Action class either as Controller Layer

resource (or) Model Layer resource as discussed above.

6) Action Forwards (Controller Layer)

→ ActionForwards are the xml entries in struts configuration file mapping ViewLayer resources (like Jsp programs) as result Pages of Struts Action class. The results gathered (or) generated by Struts Action class will be be Formatted in the View layer resources based on the ActionForwards Configuration.

7) Struts Configuration File :- (Controller Layer)

→ <any filename>.xml can act as struts configuration file. If no filename is specified explicitly then ActionServlet looks to take struts-config.xml of WEB-INF folder as default Struts Configuration.

→ The configurations done in Struts configuration file will be used by framework to generate the integration logic of Controller Servlet dynamically.

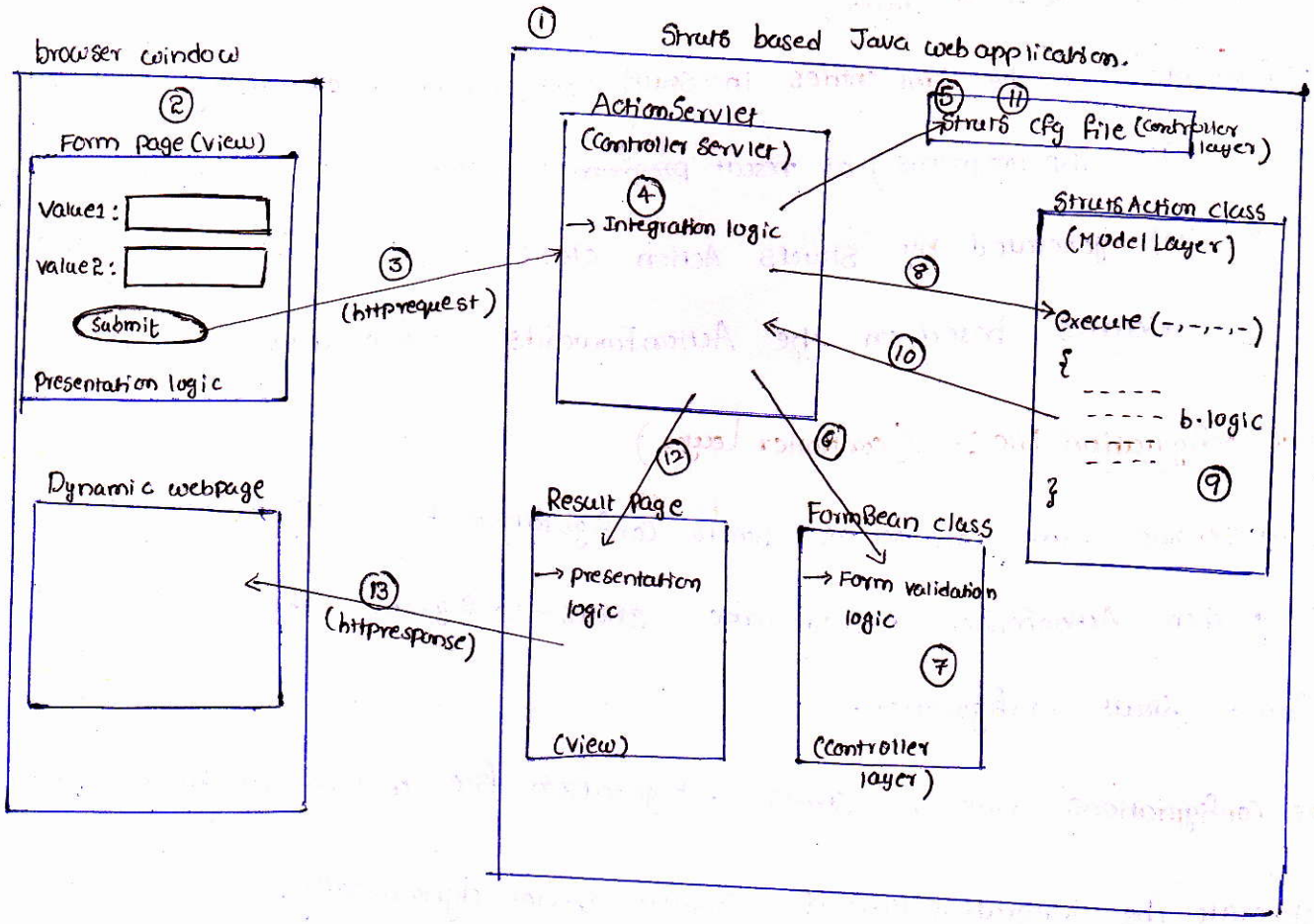
→ This file contains following configurations

*) Formbean classes cfg
*) action classes cfg
*) Action forward cfg
*) plug-ins cfg and etc...

NOTE:- In one struts application we can have multiple JSP prgs, form bean classes, action classes, struts configuration files but we must have one web.xml file and one ActionServlet Configuration.

09/11/2012

Flow of execution in Struts 1.x application :-



→ In the above diagram Business Logic is directly placed in Struts Action class. So, It comes under Model layer resource.

→ Struts is designed based on MVC2 Architecture. So, it can be used only to develop MVC2 architecture based web application.

→ The flow of execution blw the resources of Struts application is fixed. but logics in that resources are programmer choice logics. (Except Actionservlet).

W.r.to the diagram

(1) Programer deployes struts application in web server or Application server. In this process some initialization process will take place.

(2) End user launches form page of the struts application in the browser window

(3) End user submits the request to struts appleation from form page hewing form data.

(4) Action Servlet traps and takes the request as Controller

(5) Action Servlet uses the entries of Struts configuration file to decide the form bean and Action class that are required to process the request given by form page.

(6) Action Servlet creates (or) locates formbean class object and write the received formdata of form page to it.

(7) The FormValidation logic of FormBean class validates the formdata.

(8) ActionServlet creates or locates Struts ActionClass object and calls execute(-,-,-,-) method on that object.

(9) & (10) This execute(-,-,-,-) method process the request and sends control and results to ActionServlet.

(11) ActionServlet uses ActionForward configurations of Struts configuration file to decide the Resultpage of Struts Action class.

(12) ActionServlet passes the results to result page.

(13) The presentation logic of Result page formats the Results and sends HttpResponse to browser window in the form of Dynamic Webpage.

Naming Conventions while developing Struts Application :-

if Form page name is X.jsp then
take XForm as FormBean class name
XAction as Action class name

Eg:- if Form page name is register.jsp then
take RegisterForm as FormBean class name
RegisterAction as Action class name.

Recommended name for struts cfg file is: struts-config.xml

→ Design patterns are set of rules which come as best solutions for reoccuring problems of software apps/projects development.

→ The Worst Solution for reoccuring problems is called Anti-pattern.

→ Design patterns are best practices to use s/w Technologies effectively in projects development

→ ISO maintains both Design patterns and Anti patterns.

MVC2 is released as Design pattern but it can also be called as 'Architecture'. Because it has become industries Defacto standard to develop web applications and to create web frame-work s/w's like Struts, JSF and etc....

→ Struts s/w is designed based on Servlets, JSP, Javabeans, Resource bundle (properties file), Xml, Jakartha Common packages and etc.... To use this technology effectively some Design patterns are implimented in struts s/w creation.

→ some imporrent **Design patterns** are :-

(Built-in Design patterns in struts)

① Front Controller (Action Servlet)
② MVC2 (Whole struts App)
③ V.O./D.T.O class (FormBean class)
④ Abstract Controller (RequestProcessor class)
⑤ IOC/Dependency Injection (ActionServlet and FormBean)

→ Understanding different types of URL patterns :-

According to Servlet specification we can take 3 types of URL patterns

① Exact match URL pattern
② Directory match URL pattern
③ Extension match URL pattern.

① Exact match URL pattern :-

→ url pattern must begin with "/" symbol
→ url pattern must not contain "*" character

Eg:- <url-pattern> /test1 </url-pattern>.

Example request urls from browser window

http://localhost:2020/TestApp/test1 (valied)

↳ web Application name

http://localhost:2020/TestApp/abc (invalied)

/TestApp/test1/abc (invalied)

/TestApp/abc/test1 (invalied)

/TestApp/test1.do (invalied)

Other examples on exact match url patterns

1) <url-pattern> /test1/abc </url-pattern>

2) <url-pattern> /test1.do </url-pattern>

3) <url-pattern> /test1/abc/x.y </url-pattern>

NOTE-1) urlpattern hides Technology name and webresource program name from End-users.
↳ like servletclass name

(2) Directory match Urlpattern :-

→ must begin with "/" symbol.

→ must end with "*" symbol

Eg:- <url-pattern> /x/y/* </url-pattern>

Example request urls from browser window

http://localhost:2020/TestApp/x/y/abc (valied)

/TestApp/x/y/x.c (valied)

/TestApp/x/y (valied)

/TestApp/a/b/x/y/abc (Invalied)

/TestApp/Y/x/abc (Invalied)

/TestApp/x (Invalied).

Other Example Directory match url-pattern

Eg:-1 <url-pattern> /test1/test2/* </url-pattern>

Eg:-2 <url-pattern> /abc.do/test1/* </url-pattern>

Eg:-3 <url-pattern> /x/* </url-pattern>

Eg:-4 <url-pattern> /a/b/c/* </url-pattern>

10/11/2012

③ Extension match url-pattern :-

→ url pattern must begin with "*" symbol.

→ url pattern must have one extension letter or word.

Syntax :- *.<Extension>

Eg:- <url-pattern> *.do </url-pattern>

Example requests urls from browser window

http://localhost:2020/TestApp/abc.do (valied)
/TestApp/x/y/x.do (valied)
/TestApp/abc/x.c/y.do (valied)
/TestApp/abc/x/y/z.do (valied)
/TestApp/.do (valied)
/TestApp/x.c (invalied)
/TestApp/abc.do/xyz.cpp (invalied)

Other example extension match url patterns :-

Eg:- ① <url-pattern> *.cpp </url-pattern>
<url-pattern> *.c </url-pattern>
<url-pattern> *.sathya </url-pattern>

We can't form url-pattern by mixing-up multiple styles.

<url-pattern> /x/y/*.do </url-pattern>
↳ invalied pattern formation

<url-pattern> /abc.do </url-pattern>
↳ Exact match url-pattern

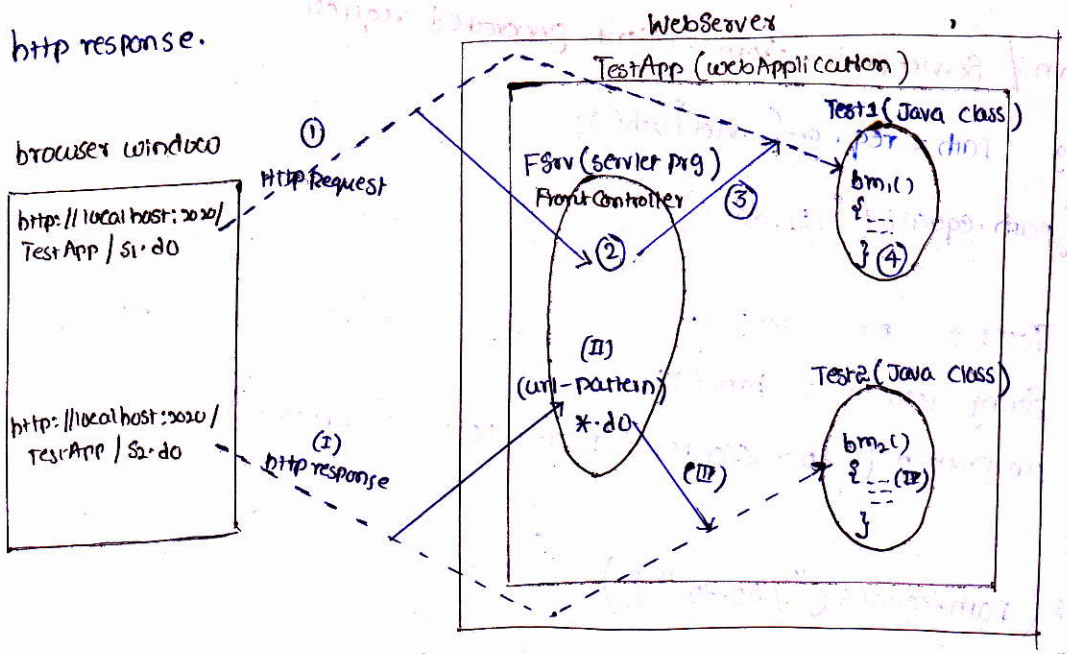
Front Controller :-

→ The special web resource program WebApp that acts as Entry and Exit point for all request and responses is called "FrontController".

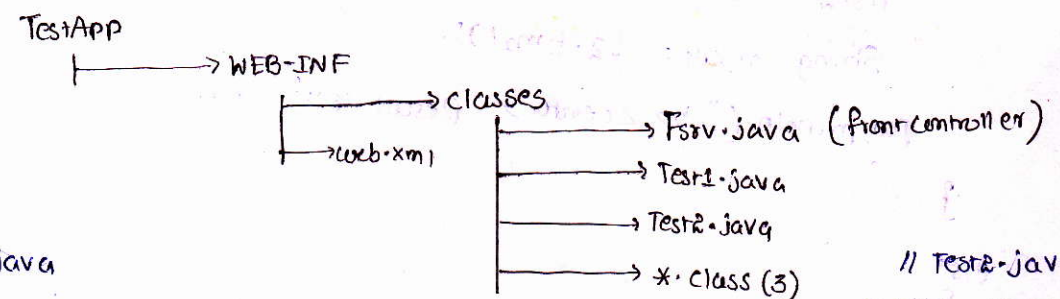
→ The Servlet | Jsp program that contains directory match (or) Extension match url-pattern can act as Front Controller.

→ The Java classes of web application can't take http request directly from clients and can't generate http responses directly back to clients. So, to take this http request from clients and to pass them to java classes we take FrontController web resource as mediator.

→ This Front Controller also gathers results from Java classes and sends to clients as http response.



Deployement Directory Structure for TestApp (classic webApp)




```
//Test1.java
public class Test1
{
  public String bm1()
  {
    return "Test1: bm1() method";
  }
}
```

```
// Test2.java
public class Test2
{
  public String bm2()
  {
    return "Test2: bm2() method";
  }
}
```

```
// FSrv.java (Front Controller Servlet prg)

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class FSrv extends HttpServlet
{
Public void doGet(HttpServletRequest req, HttpServletResponse res) throws SE, IOE
{
    //get PrintWriter
    PrintWriter pw=res.getWriter();
    // set response Content type
    res.setContentType("text/html");

//read url pattern / Servlet path from client generated request
    String path = req.getServletPath();
    if(path.equals("/s1.do"))
    {
      Test1 t1 = new Test1();
      String result = t1.bm1();
      pw.println("<b> <center> Result is ::: "+result);
    }
  else if(path.equals("/s2.do"))
    {
      Test2 t2 = new Test2();
      String result = t2.bm2();
      pw.println("<b> <center> Result is ::: "+result);
    }
  else
    {
    pw.println("<b> <i> invalied Result </i> </b>");
    }
    pw.close();
}// doGet(-,-)
```

```
public void doPost (HttpServletRequest req, HttpServletResponse res) throws SE, IOE
{
    doGet(req, res);
}
}
```

web.xml

```
<web-app>
  <servlet>
    <servlet-name> abc </s-n>
    <servlet-class> FSrv </s-c>
  </servlet>
  <servlet-mapping>
    <servlet-name> abc </s-n>
    <url-pattern> *.do </u-p>
  </s-m>
</web-app>
```

*.do → extension match url-pattern.

1> javac *.java

~~2> java~~

Request URLs: http://localhost:2020/TestApp/S1.do
S2.do

→ In Struts appn's "ActionServlet" should be taken as "FrontController Servlet", for this we need to configure ActionServlet in web.xml file either with directory match url pattern (or) Extention match Url pattern
└recomended

Q:- What is need of taking ActionServlet as FrontController ?

Ans:- in Struts application Struts Action Classes are ordinary Java classes. so, they can't take HttpRequest directly from the clients. but all request coming to Struts application will target to execute the logics of Struts Action Classes. for this we make ActionServlet as FrontController Servlet. so that ActionServlet traps and takes clients generated HttpRequest and passes them to appropriate Struts Action classes based on the configurations done in Struts Configuration file.

Code in web.xml file of Struts application to configure ActionServlet

```
<web-app>
    <servlet>
        <servlet-name> action </servlet-name>
                        └ logical name
        <servlet-class> org.apache.struts.action.ActionServlet </servlet-class>
                        └ package name          └ class name
        <init-param>
            <param-name> config </param-name>
            <param-value> WEB-INF/struts-config.xml </param-value>
        </init-param>
        <load-on-startup> 2 </load-on-startup>
                          └ priority value
    </servlet>

    <servlet-mapping>
        <servlet-name> action </servlet-name>
                        └ logical name
        <url-pattern> *.do </url-pattern>
                       └ Extension match url pattern to configure ActionServlet as
                         Front Controller Servlet. (any word or letter can be taken extension).
    </servlet-mapping>

</web-app>
```

Fixed <init-param> specifying the name & location of struts configuration file.

12/11/2012

→ we gather non-technical data from end users being from Servlet program as request parameter value. (form data).

Eg:- name, age, ... etc.

→ We gather technical data from programmers through web.xml file being from Servlet programs as Servlet init parameter values (or) Context parameter values.

Eg:- JDBC driver details,

→ The Struts configuration file name and location required for the ActionServlet is technical data and should be passed by programer. so, ActionServlet expecting as Servlet init parameter (Config) value

<load-on-startup> :-

→ If <load-on-startup> is not enabled on Servlet program, Servlet Container creates object of Servlet class when it gets 1st request to servlet program. Due to this the response time of 1st request will be little bit high when compared to the response time of other than 1st request.

↳ time taken for processing the request and generating response.

→ TO over come this problem make Servlet Container to create object of Servlet class either during Server startup or during the deployement of webapplication by enabling load-on-startup on servlet program. Due to this the response time of 1st request will be reduced (minimized) and will be equaled with the response time of other than 1st request.

Classic Webapplication :-

	<l-o-s> value	<l-o-s> value	<l-o-s> value
Srv 1	1 (II)	10 (I)	5
Srv 2	0 (I)	20 (II)	5
Srv 3	4 (III)	30 (III)	5
Srv 4	10 (IV)	-1 (Ignores <l-o-s>)	5

→ server decides the orders.

Srv1, Srv2, Srv3, Srv4 are servlet programs.

The table is given w.r.to tomcat 6.x+

Since ActionServlet is imperent Controller Servlet of Struts application to minimize the response time of its 1st request and to keep its object ready before the arrival of 1st request it is the recomended to enable <load-on-startup> on ActionServlet.

→ When multiple Servlets (programs) of webapplication are enabled with <load-on-Startup> in which order they will be instantiated and initialized by Servlet Container during Server start-up (or) during deployement of web application will be decided based on their <load-on-startup> priority values.

→ high value indicates low priority.

→ low value indicates high priority.

→ -ve value will be ignored as priority value.

→ There is no special meaning for "0" from Tomcat 6.x+.

→ Prior to Tomcat 6.x '0' indicates least priority.

→ Since ActionServlet is the only one servlet of the struts based webapp'n we can give any "+ve" number or "0" as load-on-startup priority value while configuring ActionServlet in web.xml file.

Q:- What is the difference blw web.xml file and struts-configuration file.

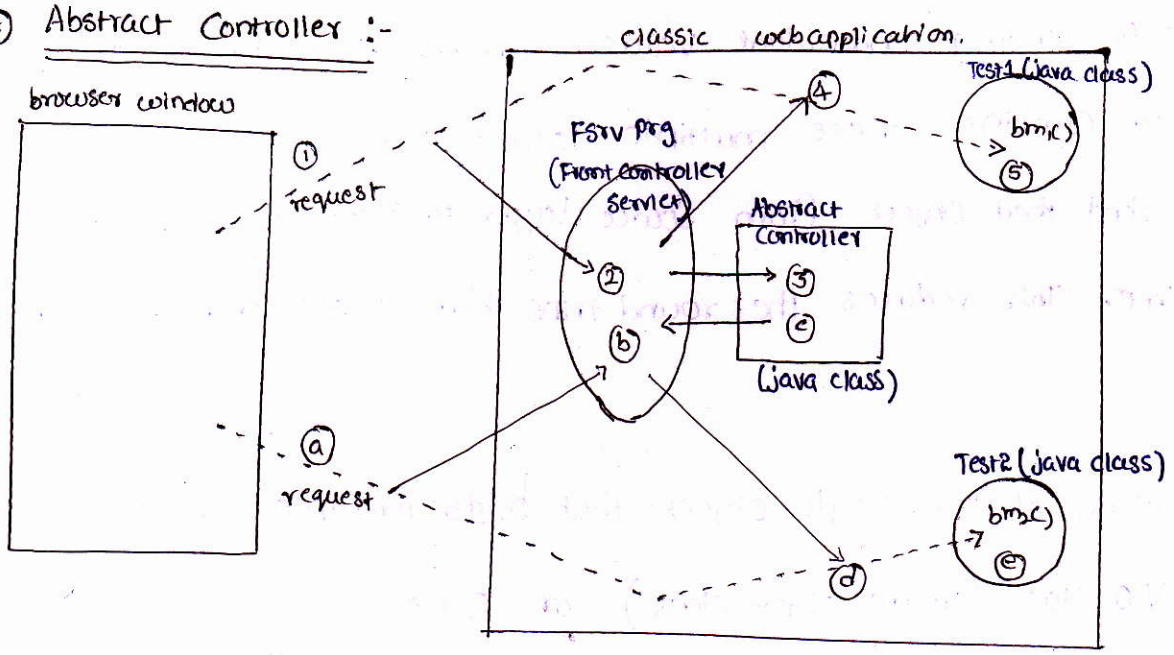
web.xml	Struts-configuration
→ It is called DD (Deployement Descriptor) file.	→ Struts Configuration file name
→ File name is fixed	→ <Any Filename>.xml
→ location is fixed (must be in WEB-INF)	→ Any location with in the wep appn
→ Servlet Container reads this file	→ ACtionServlet reads this file.
→ Contain Servlet, JSP ~~file~~ prg Configuration	→ Contain FormBeans, Actionclass, ActionFowards and etc... configurations.
→ Can be develop based on DTD (or) Schema rules.	→ Can be developed only based on DTD rules.

~~ServletContainer creates object for~~

→ Servlet Container creates object for ActionServlet.

→ ActionServlet creates objects for FormBean and ActionClasses

(x) Abstract Controller :-



→ The helper class for FrontController web resource program of webapplication is called abstract controller.

→ It is very useful to reduce the burden on FrontController and to customize the behaviour of FrontController with out touching FrontController that means we modify the code of abstract controller class to change the behaviour of FrontController

→ org.apache.struts.action.RequestProcessor (Javaclass) is abstract Controller class for the FrontController ActionServlet.

→ This RequestProcessor (c) performs lot of activities on be half of ActionServlet like FormBean, ActionClass objects creation and etc.

→ TO customize the behaviour of ActionServlet we develop Customize Request processor class. and we configure it AbstractController for ActionServlet.

(*) V.O class / D.T.O class (Value Object class / Data Transfer Object class) :- 13-04-2012

→ While copying / sending huze amount of data from source Layer to destination Layer instead of sending individual values for multiple times it is recomended to combine these multiple values into single java class Object and send that object from Source Layer to Destination Layer only for one time. This reduces the round trips b/w Source and Destination Layers.

→ Here the class of that single object that holds multiple values is called as V.O class (value object class) or D.T.O

→ In Struts applications FormBean class is called as Bean class (or) D.T.O. class. Bcz the Object of this class holds multiple values of Formpage and Becomes useful to transfer FormData from ~~Browser~~ Form Page of View layer to Struts Action Class of Controller / Model Layer as Single Object through ActionServlet.

(*) IOC (Inversion Of Control) / Dependency Injection :-

→ Dependency lookup :-

In this resource explicitly searches and gathers the dependent values from other resources of the application. In this resource pools the values from other resources

Eg:- ① If student gets his course meterial (dependent value) by requesting for it, is called dependency lookup.

Eg:- ② The way Java application uses JNDI code to get JDBC data Source object from registry is called "Dependency lookup".

→ Dependency Injection :-

In this the underlaying Container (or) F/w (or) server (or) Special resource (or) JRE dynamically assigns dependent values to the resource. In this underlaying environment Pushes the dependent values to the resource.

Eg:- ① The way Constructor executes to provide initial data to Object when Object is created.

Eg:- ② The way ServletContainer creates and assigns ServletConfig Obj to over Servlet class obj.

Eg:- ③ The way Action Servlet writes/populates form data to FormBean class obj when it receives request from the Formpage.

Eg:- If Student gets course material immediately after registring for course without any explicit request. Then it is called Dependency Injection.

(*) MVC2 (Model View Controller layers) :-

→ Developing web application by utilizing multiple technologies having multiple layers and multiple rules implementation is nothing but developing MVC2 architecture based web application.

→ Using Struts we can develop only MVC2 architecture Based webappn's.

Knowing about Struts Configuration file and its entries :-

→ <Any filename>.xml can be taken as Struts Configuration file.

→ In no File name is specified then the ActionServlet looks to take /WEB-INF/ Struts-Config.xml file as default Struts Configuration file.

In web.xml :-

```
<servlet>
    <servlet-name> action </s-n>
    <servlet-class> org.apache.struts.action.ActionServlet </s-c>
    <load-on-startup> 1 </l-o-s>
</servlet>
```

ActionServlet takes Struts-Config.xml file of WEB-INF folder as default struts cfg file

In web.xml :-

```
<servlet>
    <s-n> action </s-n>
    <s-c> org.apache.struts.action.ActionServlet </s-c>
    <init-param>
        <param-name> config </p-n>
        <param-value> /WEB-INF/mycfg.xml </p-v>
    </init-param>
    <load-on-startup> 1 </l-o-s>
</servlet>
```

ActionServlet takes mycfg.xml of WEB-INF folder as Struts cfg file.

→ The Java Bean class the extends from org.apache.struts.action.ActionForm (C) is called FormBean class.

→ The member Variables of this class are called FormBean properties. Bcz they hold Form Components data when ActionServlet writes the

FormData of FormPage to FormBean class object.

→ These FormBean property names and count should match with FormComponents names and count of FormPage.

→ Every FormBean class must be configured in Struts Configuration file having logical name. and that FormBean class is identified based on its logical name through out the struts application. ActionServlet internally uses this logical name as object name when it creates object for FormBean class.

E.g:- RegisterForm.java (FormBean)

```
package P1;
public class RegisterForm extends org.apche.struts.action.ActionForm
 {
    //FormBean properties
      String username, password;
    public void setUsername (String uname)
      {
        this.Username = uname;
      }
    public String getUsername()
      {
        return Username;
      }
    public void setPassword (Strid pwd)
      {
        this.Password = pwd;
      }
    public String getPassword()
      {
      return Password;
      }
 }
>javac -d . RegisterForm.java
```

In struts cfg file

Form Bean Configuration

```
<form-beans>
    <form-bean name="rf" type="P1.RegisterForm" />
</form-beans>
```

"rf" → FormBean logical name

"P1.RegisterForm" → Fully qualified java class acting as FormBean.

14|11|2012

→ The Java class that extends from org.apache.struts.action.Action (c) is called Struts Action Class.

→ This class must override execute(-,-,-,-) method of predefined Action class. This method returns ActionForward class object pointing to result page.

→ Every Struts Action class must be configured in Struts Configuration file having "Action Path." struts Action class will be identified through its ActionPath

→ The process of configuring Struts Action classes in Struts Configuration file by specifying there FormBeans, Result Pages (ActionForwards) is called Action mapping operation.

→ Use <form-bean> for formbean class cfg.
Use <action> for struts Action class cfg.
Use <forward> for Action forward cfg (result page)

~~Example Scenario~~

~~In struts cfg file~~

~~<struts-config>~~
~~<form-beans>~~

RegisterAction.java (Struts Action class)

```
Package P1;

Public class RegisterAction extends org.apche.struts.action.Action..
{
    Public ActionForward execute(-,-,-,-)
    {
        ------
        ------
        ------
        ------
    return ActionForward Obj (pointing to result page)
    }//execute(-,-,-,-)
}//class
```

In struts cfg file

```
<struts-config>
    <form-beans>
        <form-bean name="rf" type="P1.RegisterForm" />
    </form-beans>

    <action-mappings>
        <action Path="/reg" type="P1.RegisterAction" name="rf">
            <forward name="success" path="/result.jsp" />
        </action>
    </action-mappings>
</struts-config>
```

[Struts Action class Configuration]

[Formbean configuration]

[Action Forward cfg]

① "rf" is FormBean logical name

② "P1.RegisterForm" is Fullyqualified FormBean Class name

③ "/reg" is Action path of Struts Action class.

④ "P1.RegisterAction" is Fullyqualified Struts Action class name.

⑤ "rf" is FormBean logical name (used here to link formbean with Action class)

(6) "success" is ActionForward cfg logical name.

(7) "/result.jsp" is The result page of struts Action class.

Under Standing the code based flow of execution :-

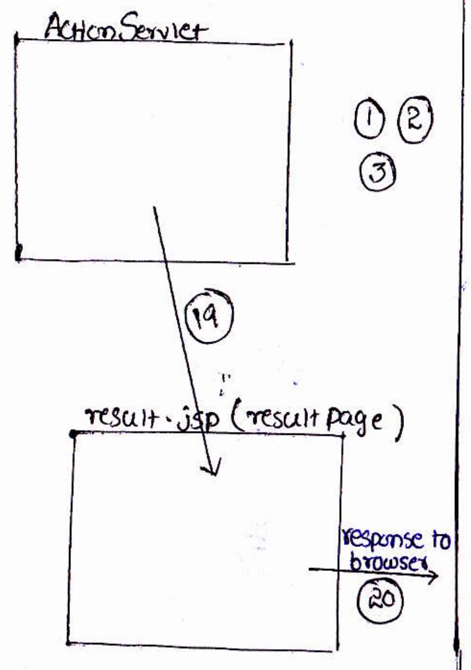
register.html (form page) (4)

```
<form action = "/reg.do" method = "get" >     (6) /reg.do -> action url
  username : <input type="text" name="uname"/><br>     (uname -> form Component name)
Password : <input type="text" name="pwd"/> <br>
  <input type="submit" value="send" /> (5)
</form>
```

In html tags based form page of struts app the action url (action attribute value of <form>) must be

<action path of struts action class> . <extension word of ActionServlet url Pattern>

↓ /reg   ↓ do



RegisterForm.java (Form bean)

```
Package P1 ;
public class RegisterForm extends
                      ActionForm
{
  // Form bean properties
private String uname ;   } must match
Private String pwd ;     } with FormComponent names
  //Write getXxx() & setXxx(-) methods
  Public void setUname(String uname)
  { - - -
    this.uname = uname;            (14)
  }
  Public String getUname ( )
  {
    return uname;
  }
  public void setPwd ( String pwd )
  {
    this.pwd = pwd ;               (14)
  }
  Public String getPwd ( )
  {
    return pwd;
  }
}
```

→ The Xxx( ) part of getter & setter methods in form bean class must match with Form Component names. Since we generally take this Xxx based on Form bean Property names, so, we can generally say Form Bean property names must match with FormComponent names.

Web.xml

```
<web-app>
  <servlet>
    <servlet-name> action </s-n> (9)
    <servlet-class> org.apache.struts.action.ActionServlet (10) </s-c>
    <load-on-startup> 1 </l-o-s>
  </servlet>
  <servlet-mapping>
    <servlet-name> action </s-n> (8)
    <url-pattern> *.do </u-p> (7)
  </servlet-mapping>
</web-app>
```

Struts-config.xml

```
<struts-config>
  <form-beans> ⟵ (13) ⟶
    <form-bean name="rf" type="P1.RegisterForm" />
  </form-beans>
  <action-mappings>
    <action path="/reg" (11) type="P1.RegisterAction" (15)
            name="rf" (12)>
      <forward name="success" path="/result.jsp" /> (18)
    </action>
  </action-mappings>
</struts-config>
```

RegisterAction.java (ActionClass)

```
Package P1;
Public class RegisterAction extends Action
{
  public ActionForward execute(-,-,-,-)
  {
    ----
    ---  (16)
    ---
    -  -
  returns ActionForward obj (having "success") (17)
  }
}
```

ActionForward logical name

15/11/2012

W.r.to the above code

① Programmer deployees struts application in web server (or) Application Server.

② Because of <load-on-startup> Servlet container creates Object of ActionServlet class either during Server startup or during the deployment of webapplication

③ In this process ActionServlet locates the struts configuration file and reads, verifies the entries of struts configuration file.

④ End-user launches Form page of struts application on Browser window. (Register.html)

⑤ End-user Fills up the Formpage and Submits the request.

⑥ Based on Action "uri" (/reg.do) placed in Form page the request url will be generated

request uri: http://localhost:2020/StrutsApp/reg.do
↳ webappn name

⑦⑧⑨⑩ Based on the Configurations done in web.xml file the ActionServlet traps and takes the request (Because request url contain reg.do and ActionServlet Url Pattern is *.do) In this process ActionServlet reads the Formdata given by Formpage.

⑪ ActionServlet uses the "reg" word of reg.do (Action Uri) and success for an Action class Configuration in Struts Configuration file whose ActionPath is "/reg". and gets the Register Action Class configuration.

⑫ By using name="rf" attribute of <Action> the ActionServlet gets the FormBean ↳ RegisterForm i.e. linked with Action Class

⑬ ActionServlet creates/locates FormBean class Object

⑭ ActionServlet calls setXxx(-) methods on FormBean class object through write the received Form data of form page to FormBean class object

⑮ ActionServlet creates (or) locates Struts ActionClass Object (RegisterAction)

⑯ ActionServlet calls execute(-,-,-,-) method on Struts Action class object to process the request by executing B.logic.

(17) execute(-,-,-,-) Creates & returns ActionForward object having logical name (success). back to ActionServlet.

(18) ActionServlet uses this logical name and ActionForward configurations done in Struts configuration file to get the result page of Action class.
↳ Result.jsp

(19) ActionServlet passes the control to result page (Result.jsp)

(20) The presentation logic of Result page formats the Result and sends response to browser window in the Form of Dynamic webpage.

*) Understanding Jsp tag libraries:-

→ Jsp tag libraries is a library. that contains set of Jsp tags.

→ The java class that extends from javax.servlet.jsp.tagext.TagSupport (C) and contains functionality of jsp tags is called tag handler class.

→ Every jsp tag library contains one tld file having the discription about Various Jsp tags.
↳ tag library descriptor.

Procedure to develop and use Custom Jsp tag library :-

Step1:- Design Jsp tag library

Step2:- Develop tag handler classes for Jsp tags in WEB-INF\classes folder

ABCTag.java (Tag handler class for <ABC>)

```
public class ABCTag extends TagSupport
{
    ------ (Gn)
    - - - - -
}
```

XYZTag.java (Tag handler class for <XYZ>)

```
Public class XYZTag extends TagSupport
{
    ----
    ----
    ---
}
```

Step-3:- Develop tld file having configuration of Jsp tags.

Sathya.tld (WEB-INF folder)
(xml code)

<ABC> ←(F)→ ABCTag.class

<XYZ> ←→ XYZTag.class

note:- tld file also contains details about sub tags, attributes and etc...

Step 1 to step3 indicates the Development of Jsp tag Library.

Step 4:- Configure Jsp tag library in web.xml having taglib uri.

In web.xml file

```
<taglib>
   <taglib-uri> demo </taglib-uri>        (D)
                  └ taglib uri
   <taglib-location> /WEB-INF/sathya.tld </taglib-location>
                                  (E)
                     └ name and location of tld file
</taglib>
```

Step-5:- Use the Jsp tags of jsp taglibrary in jsp prgs of Webapplication.

Test.jsp

```
<%@taglib uri="demo" prefix="st" %>        (C)
          (B)
<st:ABC />
          (A)
<st:XYZ />
```

Step4 and step5 indicates the utilization of Jsp tags

→ (H) output of <st:ABC> given by ABCTag class goes to browser console

→ The prefix of Jsp tag library is "user define".

→ This prefix is useful to differentiate two tags (Jsp tags) when they have got same name belonging to two different tag libraries.

→ Here "A" to "H" indicates the flow of Jsp execution.

→ To help Struts programmers Struts also supplies 5 no.of custom jsp tag libraries. Using these tags we can develop the jsp programs of Struts App as javaless jsp programs.

jsp Taglibrary	tld filename	recommended prefix	jar files having tld files, tag handler classes
html	Struts-html.tld	html	Struts-taglib-1.3.8.jar
bean	Struts-bean.tld	bean	" "
nested	Struts-nested.tld	nested	" "
logic	Struts-logic.tld	logic	" "
tiles	Struts-tiles.tld	tiles	Struts-tiles-1.3.8.jar

→ The Struts supplied html tag library is Jsp tag library containing ser of Jsp tags as aulternate for traditionalhtml tags.

→ Struts-taglib-1.3.8 jar file, Struts-tiles-1.3.8.jar files are available in <Struts_home>\lib folder.

16/11/2012

*) Procedure to use Struts supplied html tag library in the Jsp programs of Struts application

Step-I :- Gather info about the Jsp tags of Html tag library.

use Struts1.x_home\docs\struts-taglib\Index.html file

Step-II :- Gather Struts-html.tld file from struts_home\src\taglib\src\main\resources\META-INF\tld folder and place it WEB-INF folder of Struts application.

Step-III :- place the Html tag library related jar files and its dependent jar files in WEB-INF|lib folder of struts application.

Struts-taglib-1.3.8.jar (<struts_home>\lib folder).

↳ this jar file and its dependent jar files available in the struts_home\lib folder

Step-IV :- Configure Html taglibrary in web.xml file

in web.xml

```
<taglib>
    <taglib-uri> demo </taglib-uri>
                  └ taglib uri
    <taglib-location> /WEB-INF/ Struts-html.tld </taglib-location>
                       └ name and location of tld file
</taglib>
```

Step-I :- Use the jsp tags of html tag library in Jsp programs of Struts application.

Test.jsp

```
<%@ taglib uri="demo" prefix="ht" %>
    <ht:form action="reg.do" method="get">
                  └ action path of ActionClass   └ optional
        Username : <ht:text property="username" /> <br>
        password : <ht:password property="password" /> <br>
        <ht:submit value="Login" />
    </ht:form>
```

NOTE:- The Struts Supplied Jsp tags must be used only in Struts based web applications.

Q:- What is the difference blw Traditional html tags and Struts Supplied html tag library tags

Traditional html tags	Struts Supplied html tag library tags
→ Given by W3C (world wide web consert)	→ given by Apache foundation
→ tags are not case sensitive and strictly typed	→ Tags are casesensitive and strictly type
→ Does not give default Support to implement MVC2 rules	→ gives default Support
→ Can be used in any Technology based web appn including Struts	→ Can be used only in Struts applications
→ The default request method of form page is "GET"	→ The default request method of form page is "POST"
→ Cannot enhance given Action url as required for Struts application	→ can enhance.

Traditional html tags based form page for Struts application

```
<form action="reg.do" > // when ActionServlet url pattern is *.do
                └ manditory
    ------
    ------
    ------                  // default request method is "GET"
</form>
```

Struts supplied html taglibrary tags based form page for Struts Application

```
<%@ taglib uri="demo" prefix="html" %>
<html:form action="reg.do">
        .....
        .....
        .....
</html:form>
```

optional

// default method in this "post"

→ If form page is design by using Traditional html tags then enabling "load-on-startup" on ActionServlet is "optional."

First page

→ If same form page is design by using Struts supplied JSP tags then enabling "load-on-Startup" on ActionServlet is "manditory".

First page

*) The two overloaded forms execute(-,-,-,-) method

→ Form1:

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                             ServletRequest request, ServletResponse response)
```

→ Form2:

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                             HttpServletRequest request, HttpServletResponse response)
```

It is recomended to use Form2 version of execute(-,-,-,-) method

ActionMapping mapping :-

Using this Object we can gather information from ActionServlet, RequestProcessor class regarding the current request coming to current Action class. This Object also useful to return ActionForward obj from execute(-,-,-,-) to ActionServlet. This Object can be used to make one Action class interacting with another Action class.

ActionForm form :- The ActionServlet supplied FormBean class obj to Struts Action class is nothing but this 2nd parameter of execute(-, ActionForm form, -, -) method,

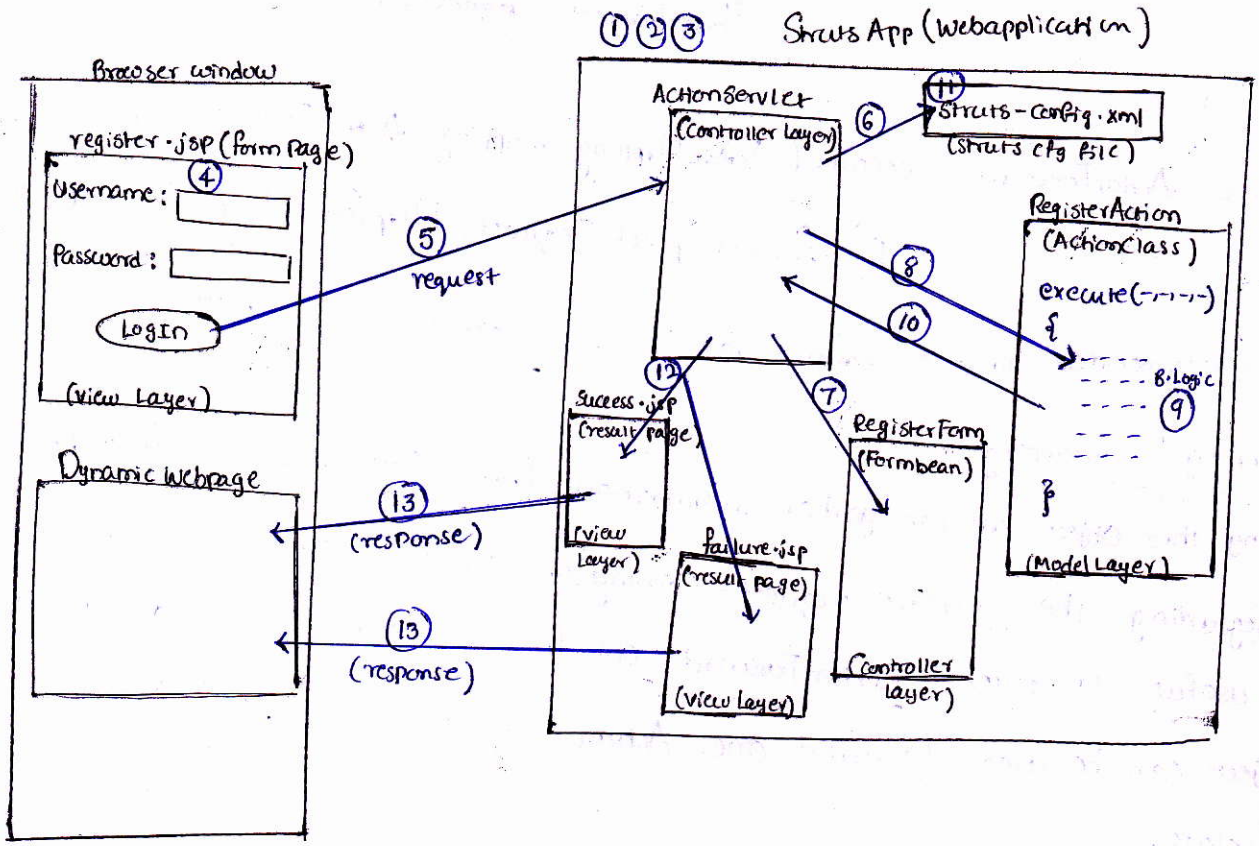
HttpServletRequest request :- using this object we can gather all the details from client generated request like misceleneous info, request header values,

reques body values (Form data) and etc..... Using this Object we can create "request attributes" to pass data from Action class to Other resources.

NOTE: Eventhough we can use request object to gather form data it is recomended to use FormBean class object to gather the Form data.

HttpServletResponse response :- If StrutsActionClass wants to send direct response to Browser window then we can do response related settings in ActionClass by using this "response" object. like Setting Content type, getting PrintWriter object, Setting ResponseHeader values and etc... This is less utilized parameter of Execute(-,-,-,-) method. Because Sending response to browser window directly from Action class is against of (violating) MVC2 rules.

1st Application :-



for the source code of this application refer page no's (16) & (17) of the 1st App in Booklet

17/11/2012

→ When Superclass reference variable refers to it subclass object then using that reference variable we can't call direct methods of subclass for this we need to type cast this Super class reference variable with subclass.

Eg:-

```
Class Test
{
  public void x()
  {
    ≡
  }
}

Class Abc extends Test
{
  public void x()
  {
    ≡
  }
  public void Y()
  {
    ≡
  }
}

Test t=new Abc();
t.x(); // valied
t.Y(); // Invalied

Abc ab=(Abc) t;
ab.x(); // valied
ab.Y(); // valied
```

→ The execute(-,-,-,-) method supplies ActionForm class reference variable (form) pointing to our FormBean class object (nothing but subclass object of ActionForm class) So, to call direct methods of our FormBean class using that reference variable "typecaste" that reference variable with our FormBean class

Eg:-

```
ActionForm form = new app.RegisterForm();
    Form.getUsername(); //invalied
    Form.getPassword(); //invalied

app.RegisterForm fm = (app.RegisterForm) form;
    fm.getUsername(); // (valied)
    fm.getPassword(); // (valied)
```

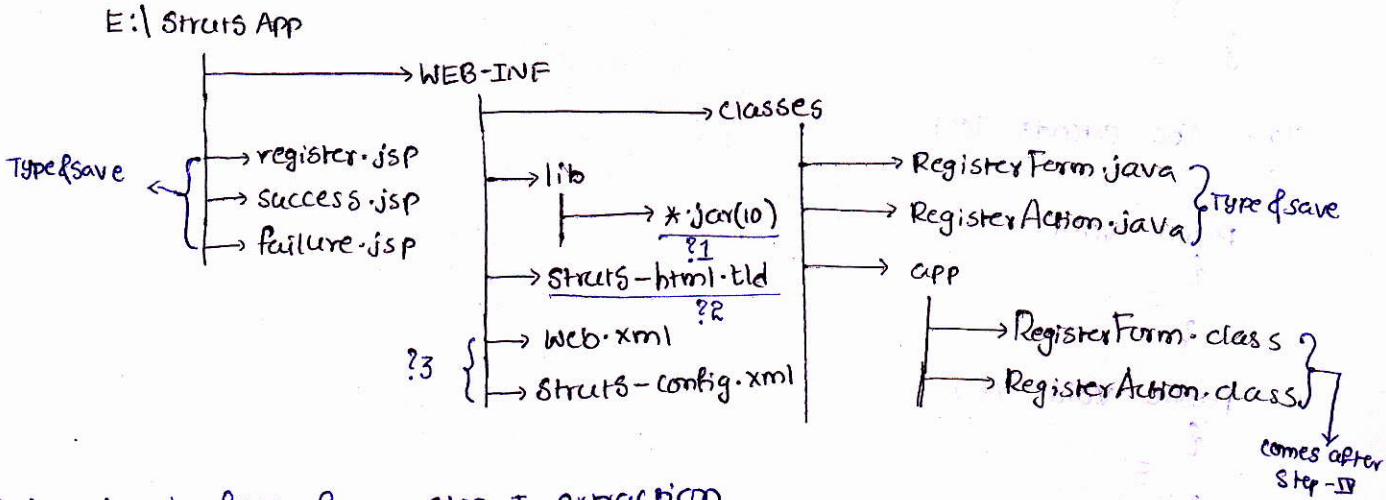
Procedure to develop, deployee and execute 1st struts application

19/11/2012

(refer page No's 16 & 17)

Step-I:- Extract <Struts_home>\apps\Struts-blank-1.3.8.WAR file to temp folder by using Winzip tool (WinRAR

Step-II:- create deployement directory structure of Struts-based web application.



?1) gather the jar files from Step-I extraction.

?2) gather Struts-html.tld file from

<Struts_home>\src\taglib\src\main\resources\META-INF\tld folder

?3) copy from Step-I extraction and modify as required

Step-III:- add tomcat_home\lib\servlet-api.jar (Represents Servlet-api),

Struts_home\lib\Struts-core-1.3.8.jar (Represents struts-api) files to CLASSPATH environment variable.

In MyComputer properties

Variable name: CLASSPATH

Value : E:\Struts 1.x soft\Struts-1.3.8\lib\Struts-core-1.3.8.jar (<Struts_home>); D:\TomcatGo\ (<tomcat_home>) lib\servlet-api.jar; <other values>;.

Step-IV:- Compile the .java resources of Struts application.

E:\StrutsApp\WEB-INF\classes> javac -d . *.java

E:\StrutsApp\WEB-INF\classes>

Step-V:- Start the tomcat server

Deployee the above Struts application in Tomcat server.

Copy E:\StrutsApp folder to tomcat-home\WEBAPPS folder

Step-VI :- Test the webapplication.

http://localhost:2020/StrutsApp/register.jsp

→ Servlet Container uses the xml parser SAX (Simple API for Xml) to read and process Web.xml. Similarly ActionServlet uses its own copy of SAX parser to read and process Struts configuration file.

→ Xml parser is a s/w application that validates the xml documents against DTD/schema rules and also process the xml document.

→ These xml parsers read the data from xml files and stores them in collection Data Structures to make that data visible through out the application.

→ modifications done in Struts configuration file will be reflected only after reloading of the web application (StrutsApp)

→ modifications done in .java files of the Struts application will be reflected after recompilation of .java files and reloading of the webapplication

→ Servlet Container collects ActionServlet class from WEB-INF/lib/struts-core-1.3.jar file of deployeed Struts application.

The s/w setup that is used to develop the above Struts app

jdk 1.4+ (jdk 1.6)

Struts 1.x (Struts 1.3.8)

Tomcat 5.x+ (tomcat 6.0)

→ If Struts configuration filename struts-config.xml and WEB-INF respectively. then that file name and location must be specified as "config" init parameter value during ActionServlet Configuration done in web.xml file.

→ ActionServlet uses 0-param constructor while instantiating formbean, Struts Action class object. so, we must makesure that 0-param Constructor is available in FormBean class and Action class directly (or) indirectly.

(*1) → explicitly placed

→ The compile generated default Constructor

→ Compiler generates 0-param Constructor as default Constructor during compilation, When no explicitly placed Constructors are there in java class.

Q:- Can I place only paramcterized constructors in FormBean, Struts Action classes ?

Ans:- not possible, because when class contains userdefined constructors, the compiler does not generate 0-param constructor as default constructor. But ActionServlet always looks to use 0-param Constructor to instantiate FormBean, Struts Action classes

→ To make our FormBean, Actionclasses visible to ActionServlet we must take them as "public" classes Because only "public classes" are visible inside and outside the package.

→ Creating (or) locating formbean class object is the responsibility of ActionServlet

→ When struts supplied Jsp tags based Form page is launched on the browser window as first page-following operations takes place.

*** ① ActionServlet creates (or) Locates the FormBean class object related to FormPage

*** ② ActionServlet calls getXxx()methods on that object (FormBean object) and writes received values to as the initial values of FormComponents in FormPage.

NOTE:- to do all these operations ActionServlet class object must be ready before launching of Form Page / First page. forthat we must enable <load-on-startup> on ActionServlet. This indicates enabling <load-on-startup> on Action Servlet is manditory when FirstPage / Form page of the Struts Application these designed by using struts supplied Jsp tags.

*** → When traditional HTML tags based formpage is launch as 1st page on the browser window Action Servlet need not to perform any activities on FormBean. so, there is no need of & having Object of ActionServlet before launching Formpage / First Page

This indicates enabling <load-on-startup> on ActionServlet is optional When

Form page / First page is designed by using traditional html tags.

→ Struts supplied Jsp tags always work in co-ordination with ActionServlet and Struts framework. so, we must keep ActionServlet class object ready before execution of these Jsp tags.

for this enabled <load-on-startup> on ActionServlet.

→ Traditional Html tags worked independently with out having any co-ordination with ActionServlet and Struts framework. so, there is no need of keeping ActionServlet obj ready before executing these Html tags.

→ we can make ActionServlet to keep FormBean class object in two different scopes.

① "request" Scope

→ FormBean class object is specific to each request

→ ActionServlet creates one new FormBean class obj for every request generated by Form page

② "session" scope

→ FormBean class object is specific to client (Browser window)

→ ActionServlet creates FormBean class object on one per browser window (client) basis

21/11/2012

Code to specify the form bean scope :-

In struts-config.xml

```
<form-beans>
    <form-bean  name="rf"  type="app.RegisterForm" />
</form-beans>

<action-mappings>
    <action  path="/register"  type="app.RegisterAction"  name="rf"  scope="request/session">
        ------
        ------
        ------
    </action>
</action-mappings>
```

name → FormBean logical name

scope → FormBean scope (request / session; default)

NOTE: To control form bean scope with r.s.to the ActionClass that is using FormBean the "scope" attribute is given in <action> to specify the form bean scope.

NOTE:- Multiple Action classes of Struts App can use Single FormBean but Single Action class can not use multiple FormBeans.

→ There is no provision to specify Scope for Struts Action class object Becoz ActionServlet creates/locates this object based on the availability of Object. That means ActionServlet locates and uses existing object of Struts Action class if it is already available. Otherwise creates and use new object

→ If U want to preserve the submitted form data of form page across the multiple request given from a browser window then keep FormBean in Session scope. other wise keep FormBean in request Scope.

The code based flow of execution of first application that is given in page nos 16 & 17 (FormBean Scope is session).

A) Programmer deploys Struts App web application in Server. The Servlet Container of Server reads and verifies web.xml file. (refline: 11-31)

B) Because of <load-on-startup> the Servlet Container creates object of ActionServlet either during Server Startup (or) during the deployment of Struts appn. (refer line no: 20).

C) ActionServlet reads and verifies the entries of Struts Configuration file and recognizes the FormBeans and Action Classes that are configured in Struts cfg uration file (refer line no's : 16-19 & 33-50)

D) Enduser launches the form page on the browser window by using the following request url

http://localhost:2020/StrutsApp/register.jsp (refer line no's : 4-10)

E) The Action URL of Formpage will be gathered (action = "register") (refer line: 6) → gathers RegisterAction class cfg done in Struts cfg file whose action

path is "/register" (refer line: 43) ⟶ gathers RegisterForm class that is linked with RegisterAction class based the formbean logical name (name="rf") refer line: 45 & 40

note: All the (E) step operations will be done by Struts supplied JSP tags in Association with ActionServlet.

F) ActionServlet creates FormBean class object and keeps that object in "Session" Scope. and also calls getXxx() methods on that object to read data from FormBean properties and writes that data/values to the Form Components of form page (text boxes) as initial values

(refer line: 40 & 69 to 76 & 7 & 8)
- 40 → FormBean Configuration
- 69 to 76 → calling getXxx(-)
- 7 & 8 → to write receive data to form components.

G) End user fills up the form page and submits the form page (refer line: 9)

H) form page generates the following request url.

http://localhost:2020/StrutsApp/register.do (refer line no: 6)

I) Since in the url pattern of ActionServlet is "*.do". The ActionServlet traps and takes the request. (23 to 26 & 13 to 21)

J) ActionServlet looks for Struts Action class configuration whose action path is "/register" (based on register.do of request url) and gets RegisterAction class cfg (refer lines: 43-48)

K) Based on name="rf" attribute of <action> ActionServlet gets RegisterForm class as form bean class that is linked with RegisterAction class (refer lines: 45 and 40)

L) ActionServlet locates RegisterForm class obj from session scope and writes received form data of Form Page to ~~FormBean~~ that obj by calling setXxx(-) on that method. (refer lines: 61 to 68)

M) ActionServlet creates ~~Process~~ RegisterAction class object (refer line no: 44)
- because first request it creates

N) ActionServlet calls execute(-,-,-,-) on RegisterAction class obj having four arguments (1. ActionMapping obj
2. FormBean class obj (RegisterForm)
3. request obj
4. response obj) (refer lines: 85 to 97)

NOTE:- This execute(-,-,-,-) reads form data from FormBean class object and uses that data as input values while executing B.logic.

Success

execute(-,-,-,-) returns ActionForward obj having logical name "success" (ref: 94) ⟶ (O) ⟵ execute(-,-,-,-) returns ActionForward obj having logicalname "failure" (ref: 96)

failure

ActionServlet uses "success" logicalname and gets "success.jsp" as result page RegisterAction class. (ref lines: (46)) ⟶ (P) ⟵ ActionServlet uses "failure" logical name and gets "failure.jsp" as result page of RegisterAction class. (ref line: (47)).

Success.jsp executes (ref 99-108) ⟶ (Q) ⟵ failure.jsp executes (109-120)

end ⟶ (R) ⟵ Clicks on Try Again hyperlink (ref: 117)

(S) ⟵ Form page will be launched (ref 4-10)

end

22/11/2012

→ There are two approaches to use Custom/Third party Jsp Tag libraries

Approach ① :- By using User-defined taglib uri

Approach ② :- By using fixed/pre-defined taglib uri

Ⓧ procedure to work with custom/Third party Jsp tag library by using Approach ① (having user-defined jsp taglib uri)

Step-① :- gather tld file and placed in webapplications deployement directory structure

Eg:- like placing Struts-html.tld file in WEB-INF folder

Step-② :- Configure Jsp tag library in web.xml file

Eg:-
```
<taglib>
    <taglib-uri> demo </taglib-uri>
                  → user-defined taglib uri
    <taglib-location> /WEB-INF/Struts-html.tld </taglib-location>
</taglib>
```

Step-③ :- place Jsp tag library related jar files in WEB-INF/lib folder

Eg:- like placing Struts-taglib-1.3.8.jar file and its dependent jar files in WEB-INF/lib folder

Step-④ :- Specify Jsp taglib uri in the Jsp programs of webapplication and use Jsp tags

Eg:- register.jsp

```
<%@ taglib uri="demo" prefix="html" %>
    ↳
<html:form action="register">
    .....
    .....
    .....
</html:form>
```

NOTE:- In our 1st Struts application Struts supplied html taglibrary is used based on approach ①.

Ⓧ procedure to work with Custom/Third party Jsp tag library by using Approach ② (using pre-defined

generally the jar files that represent Jsp tag library also contains tld files

So, there is no need of supplied tld files superately to the web application.

→ In every tld file fixed/pre-defined taglib uri is available as the content of <uri> tag

→ The fixed taglib uri's of struts supplied Jsp taglibraries are html tag library

html taglibrary	http://struts.apache.org/tags-html	(struts-html.tld)
bean taglibrary	http://struts.apache.org/tags-bean	(struts-bean.tld)
logic taglibrary	http://struts.apache.org/tags-logic	(struts-logic.tld)
nested taglibrary	http://struts.apache.org/tags-nested	(struts-nested.tld)
tiles tag library	http://struts.apache.org/tags-tiles	(struts-tiles.tld)

Procedure :-

Step-(1) :- place the Jsp tag library related jar files in WEB-INF/lib folder of web application

Eg:- like placing struts-taglib-1.3.8.jar and its dependent jar files in WEB-INF/lib folder

Step-(2) :- gather the fixed taglib uri of Jsp tag library from the <uri> tag of its tld file

Eg:- refer above table

Step-(3) :- use the tags of jsp tag library in Jsp programs by specifying fixed taglib uri

Eg:- register.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<html:form action="register">
        ------
        ------
        ------
</html:form>
```

NOTE:- Approach (2) is always recommended to use to work with Jsp taglibraries

→ When Source webresource program interacts with destination webresource program using rd.forward() then source and destination programs

use same request and response object.

→ If source webresource program interacts with destination web resource program usir response.sendRedirect() then source and destination programs will not use same request and response objects.

→ ActionServlet and Action class use same request and response objects weather ActionServlet and ResultPage use same request object or not depends upon redirect = "true/false".

Eg:- in Struts configuration file

```
<action path="/register" type="app.RegisterAction" name="rf">
    <forward name="success" path="/success.jsp" redirect="false" />
    <forward name="failure" path="/failure.jsp" redirect="true" />
</action>
```

→ redirect = "false" :- (false is default value)

ActionServlet uses rd.forward(-,-) to pass the control to resultpage. so,

a) ActionServlet and ResultPage use same req and res objs

b) Action class and its Resultpage use same req and res objs

→ redirect = "true" :-

ActionServlet uses res.sendRedirect(-) to pass the control to resultpage. so,

a) ActionServlet and resultpage will not use same req and res objects

b) Action class and its resultpage will not use same req and res objs

In the above code

ActionServlet uses rd.forward(-,-) to interact with success.jsp

ActionServlet uses res.sendRedirect(-) to interact with failure.jsp

→ request attributes are specific to each request. The source and destination programs must use same request and response objects in order to work with request attributes.

→ Session Attributes are specific to client (Browserwindow) the source and destination programs must get request from same browserwindow to work with session attributes.

→ ServletContext attribute are webapplication specific attributes. These attributes are condition less global attributes with in the web application.

Q:- How to pass the data between StrutsAction class and its result page?

Ans:- *) If Struts Action class and its resultpage are using same request and response objects then use request attributes

*) If Struts Action class and its resultpage are getting request from same browser window then use session attributes

*) If you can't create above two situations then use ServletContext attributes.

attribute name must be string and value must be an object. These attributes are no way related with HTML (or) XML tag attributes.

Example :- (w.r.to 1st struts application)

Step-I :- add following additional code in the execute(-,-,-,-) method of RegisterAction class.

```
//request attribute
req.setAttribute("attr1","val1");

//session attribute
HttpSession ses = req.getSession(true);
ses.setAttribute("attr2","val2");

//ServletContext attribute
ServletContext sc=ses.getServletContext();
sc.setAttribute("age",30);
                attribute_value
```

//Write following code in Success.jsp / failure.jsp to read attribute values

Success.jsp

```
<html>
  <body>
    <center>
      <font size=5> Login Successful </font>
    </center>
    attr1(req) value : <%= request.getAttribute("attr1") %> <br>
    attr2 (ses) value : <%= session.getAttribute("attr2") %> <br>
    attr3 (sc) value : <%= application.getAttribute("age") %> <br>
  </body>
</html>
```

failure.jsp

Same as above

```
<font size=5> Login failure </font>
```

23/11/2012

→ It is recomended to write code in Jsp programs purely through tags and without using any kind of Java statements.

→ Struts supplies <bean:write> tag to read attribute values from different scopes.

In Success.jsp / failure.jsp :-

```
<%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<html>
  <body>
    <center>
      <font size=5 color=green> Login Success </font>
    </center>
Attr1(req) value : <bean:write name="attr1" scope="request" /> <br>
Attr2 (ses) value: <bean:write name="attr2" scope="session" /> <br>
Age (sc) value: <bean:write name="age" scope="application" format="" />
  </body>
</html>
```

name → attribute name
scope → attribute scope
format → required if attribute value is numeric data.

→ "scope" attribute is optional in the above tags.

→ if "scope" attribute is not specified the pageContext.findAttribute(-) will be used to search and retrive the attribute values from multiple scopes (like pagescope, requestscope, sessionscope, Applicationscope)

→ We can use <logic:notEmpty> to check weather given attribute is there in specific scope or not by specifying scope attribute.

if scope attribute is not specified then it verifies the attribute availability in Multiple scopes.

→ Since <bean:write> tag throcos Exception if specified attribute is not available. so, it is recomended to use <bean:write> tag as the nested tag of <logic:notEmpty> tag.

```
in Success.jsp / failure.jsp
<%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<html>
  <body>
    <center>
      <font size=5 color=red> Login succes </font> <br>
    </center>

(**1) <logic:notEmpty name="attr1">
        Attr1(req) value : <bean:write name="attr1" /> <br>
      </logic:notEmpty>

  <logic:notEmpty name="attr2" scope="session">
     ATTR2 (ses) value : <bean:write name="attr2" /> <br>
  </logic:notEmpty>

  <logic:notEmpty name="age" scope="application">
     Age (sc) value : <bean:write name="age" format="" /> <br>
  </logic:notEmpty>
  </body>
</html>
```

(**1) the equalent java code in jsp is

```
<% if(pageContext.findAttribute("attr1")!=null)
   {%>
Attr1(req) value : <%= pageContext.findAttribute("attr1") %>
```

(expression tag)

<% } %>

→ We can also use <bean:write> tag to read Formdata from the FormBean class Object properties by specifying "property" attribute.

in Success.jsp / failure.jsp

```
The Form Data is:
    <bean:write name="rf" property="username" />
    <bean:write name="rf" property="passcoord" />
                  |                  |
                  |                  → FormBean Property name
                  → FormBean logical name
</body>
</html>
```

Q:- Can U explain the purpose of various Struts supplied Jsp tag libraries?

→ refer <Struts-home>\docs\Struts-taglib\index.html

→ tiles tag library is required to design the webpages having tiles (partitions) like Designing webpages having header, Footer, main-content and etc...

The Five Struts supple jsp taglibraries are:-

- html
- bean
- logic
- nested
- tiles

Q:- How to display the result generated by StrutsAction class in the FormPage it self?

Ans:- take formpage as ResultPage of Actionclass

Example:- W.r.to 1st appn.

Step 1:- configure a register.jsp as the result page of RegisterAction class in Struts-Config.xml

```
<action-mappings>
  <action path="/register" type="app.RegisterAction" name="rf">
    <forward name="result" path="/register.jsp" />
  </action>
</action-mappings>
</struts-config>
```

Step-2: Write following code in the execute(-,-,-,-) of RegisterAction class write after reading Formdata from FormBean class object.

```
String user = rf.getUsername();
String pass = rf.getPassword();
if(user.equals("sathya") && pass.equals("tech"))
  {
    req.setAttribute("msg", "Valied Credentials");
  }
else
  {
    req.setAttribute("msg", "Invalied Credentials");
  }
return mapping.findForward("result");
  }
}
```

Step-3: Write the following additional code in Result.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>

<logic:notEmpty name="msg" scope="request">
  Result is : <bean:write name="msg" scope="request"/>
</logic:notEmpty>
```

27/11/2012

→ The text file that maintains the entries in the form of "key=value" pairs is called as text properties file.

Eg:- myFile.properties

```
#text properties file
my.name = raja
my.age = 27
my.addrs = hyd
```

keys → my.name, my.age, my.addrs
values → raja, 27, hyd

→ The standard principle in slw industry is

don't hard code those values in your application which are changable in future. It is recomended to pass such values to application from outside the application by taking the support of text properties file.

→ While working with struts application the properties file must have .properties extention.

→ we can take one (or) more properties files in a struts application and all these files must be configured in struts configuration file using <message-resources> tag

*) In struts application we use properties files for the following operations.

① to maintain presentation labels and to make them flexible to modify and reusable.

② to maintain presentation label supporting i18n (internationalization)

③ to maintain FormValidation error messages.

④ to maintain styles required for presentation lables and error messages.

⑤ to maintain Jdbc properties required for Struts Actionclass

Procedure to add properties file support to our 1st Struts application.

Step 1:- Prepare properties file in WEB-INF/classes folder

```
# presentation labels
my.title = <center> <h1> Login Page </h1> </center>
my.user = Username
my.pwd = Password
my.btn.cap = Login
my.res = Result is
```

save "myfile.properties"

Step 2:- Configure properties file in Struts Configuration file

in struts-config.xml

```
</action-mappings>
<message-resources parameter = "myfile" />
</struts-config>
```

Step 3:- read the labels from properties file and use them in Formpage or jsp programs.

register.jsp

```
<%@ taglib >
<%@ taglib >
<%@ taglib >

<bean:message key="my.title" />
<html:form action="register" >
  <bean:message key="my.user" /> <html:text property="username" /> <br>
  <bean:message key="my.pwd" /> <html:password property="password" /> <br>
<html:submit>
  <bean:message key="my.btn.cap" />
</html:submit>
</html:form>

<logic:notEmpty name="msg" >
  <bean:message key="my.res" /> <bean:write name="msg" />
</logic:notEmpty>
```

→ To make messages (values) of properties file as reusable for multiple labels we can design messages with arguments. ({0} .... {4})

↳ max of 5 args

→ we can supply values to these arguments by using arg0 ..... arg4 attributes (5 attributes) of <bean:message> tag

Eg:- myfile.properties

```
#presentation labels
my.lbl = Login {0}
~~my.btn.cap = Login~~
my.res = Result is
```

→ Register.jsp

```
<center> <h1> <bean:message key="my.lbl" arg0="page" </h1> </center>
                                  ↳ supplies {0} value
<html:form action="register">
  <bean:message key="my.lbl" arg0="username" /> <html:text property=
  <bean:message key="my.lbl" arg0="password" /> <html:Password property=
<html:submit>
  <bean:message key="my.lbl" arg0="" />
</html:submit>
</html:form>

<logic:notEmpty name="msg">
  <bean:message key="my.res" /> <bean:write name="msg" />
</logic:notEmpty>
```

→ It is not recomended to keep more than 500 messages in one properties file. for better performance. To solve this problem, it is recomended to use multiple properties files.

→ while Configuring multiple properties files in struts configuration file provide logical names to them for identity using "key" attribute of <message-resources> tag.

Eg:-

Step-1:- keep multiple properties files ready.

→ myfile.properties (WEB-INF/ classes folder)

```
# presentation lables
my.title = <center> <h1> Login page </h1> </center>
my.user = UserName
my.pwd = password
```

→ myfile1.properties (WEB-INF/ classes/ app folder)

```
my.btn.cap = Login
my.res = Result is
my.footer = <i> <center> from Struts App </center> </i>
```

Step 2:- Configure the properties files in struts configuration file.

```
</action-mappings>
<message-resources  parameter = "myfile"  key = "f1" />   → logical name of myfile
<message-resources  parameter = "app.myfile1"  key = "f2" />   → logical name of app.myfile
```

Step-3:- read the labels from multiple properties file.

register.jsp

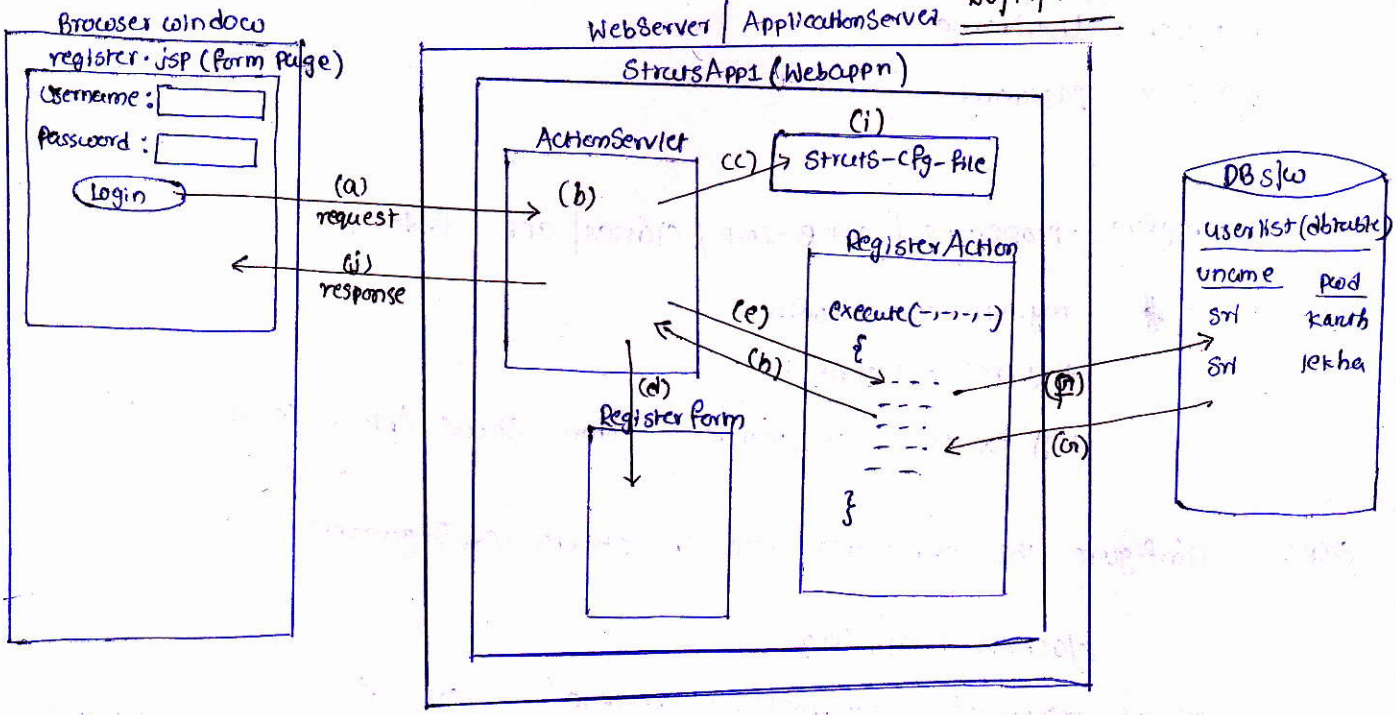
```
<center> <h1> <bean:message  key = "my.title"  bundle = "f1" /> </h1> </center>

<html:form  action = "register" >
  <bean:message  key = "my.user"  bundle = "f1" /> <html:text property=
  <bean:message  key = "my.pwd"  bundle = "f1" /> <html:password property=
  <html:submit>
    <bean:message  key = "my.btn.cap"  bundle = "f2" />
  </html:submit>
</html:form>

<logic:notEmpty  name = "msg" >
  <bean:message  key = "my.res"  bundle = "f2" /> <bean:write name="msg" />
</logic:notEmpty>
<br>
```

<bean : message key = "my.footer" bundle = "f2" />

→ Struts project created in NetBeans IDE gives one-built in property file and also allows us to add additional properties file.

* STRUTS TO DATABASE COMMUNICATION *

26/11/2012



*) for Struts application to DataBase s/w communication place persistence logic like (Jdbc code) in StrutsAction class.

procedure to develop the above appn in NETBEANS IDE (6.7.1)

StrutsDBApp → in work\struts

Create Table in DataBase :-

```
SQL> create table userlist (uname varchar2(20), pwd varchar2(20));
SQL> Insert into userlist values ("sri", "kanth");
SQL> Commit;
```

SQL query of the appn :-

```
SQL> Select count(*) from userlist where uname="sri" and pwd="kanth";

  count(*)
  --------
     1  ------> valied

SQL> select count(*) from userlist where uname="nookal" and pwd="sri";

  count(*)
  --------
     0  ------> Invalied
```

procedure :-

Step-(1):- create Java web project in NetBeans IDE

File → New project → Javaweb → webappn → next

project name : StrutsApp 1 → next → server : GlassFish 2.1.x →

Context path : /StrutsApp 1 → war file name → next → ☑ struts 1.3.8

→ Action URL pattern : *.do → ☑ add Struts flds → Finish →

Delete welcome Struts.jsp, Index.jsp from Webpages of WEB-INF Folder.

Step-(2):- add ojdbc14.jar file to the libraries of project

Rightclick on libraries → Addjar → Browse & select ojdbc14.jar

Step-(3):- Develop form page (Register.jsp)

Rightclick on webpages folder → new → jsp

Jsp file name : register → finish.

Register.jsp

```
<%@ taglib uri = "http://struts-apache.org/tags-html" prefix = "html" %>
<%@ taglib  "  "  "  /tags-bean"  "  = "bean" %>
<%@ taglib  "  "  "  /tags-logic"  "  = "logic" %>

<html:form action = "register" >
    Username : <html:text property = "username" />
    Password : <html:password property = "password" />
    <html:submit value = "login" />
</html:form>
```

Step-(4):- Add FormBean class to the project

Right click on Source packages folder → new → Others → Struts (leftside)

→ R. Struts ActionForm Bean → Class Name : Register Form → Finish

RegisterForm.java :-

```
public class RegisterForm extends org.apache.struts.action.ActionForm
{
    //FormBean properties
    private String username, password;
    //write setters & getters methods
}
```

→ select above FormBean properties → Right click → insert code → getters & setters.

Step-⑤ :- Add StrutsAction class to project

Right click on Source package (src) folder → new → other → struts → Struts Action → next → ClassName : RegisterAction → actionpath : /register → next → ActionFormBeanName RegisterForm → scope : ⊙ session. → finish.

Step-⑥ :- Configure RegisterAction class as shown below in struts configuration file

In struts-config.xml file of configuration files folder :-

```
<action name="RegisterForm" path="/register" scope="session" type="RegisterAction">
    <forward name="result" path="/register.jsp" >
</action>
```

(RegisterForm → logical name)

Step-⑦ :- Write following code in the execute(-,-,-,-) method of RegisterAction.java

```
public ActionForward execute(-,-,-,-)
{
    //read Form data from FormBean class object
    RegisterForm fm = (RegisterForm) form;
    String un = fm.getUsername();
    String pass = fm.getPassword();

    //write the jdbc code
    Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","scott","tiger");
```

cntr + shift ti for package

```
PreparedStatement ps = con.prepareStatement("select count(*) from userlist
                                              where username=? and pwd=?");
    ps.setString(1,un);
    ps.setString(2,pass);
  ResultSet rs = ps.executeQuery();
    int cnt=0;
    if(rs.next())
        cnt = rs.getInt(1);
    if(cnt==0)
      request.setAttribute("msg","Invalied Credentials");
    else
      request.setAttribute("msg","valied Credentials");
    //returns ActionForward obj
    return mapping.findForward("result");
  }//execute(-,-,-,-)
}
```

Step-⑧:- write the following additional code in Register.jsp

```
<logic:notEmpty name="msg" scope="request">
Result is : <bean:write name="msg" scope="request">
</logic:notEmpty>
```

Step-⑨:- Right click on project → run

procedure to modify the http port number of default "domain1" domain of GlassFish server.

<GlassFish-home>\appserver\domains\domain1\config\domain.xml

→ modify the port attribute value of first <http-listener> tag.

---

*) what is SQL Injection problem

*) develop appn demonstration SQL Injection problem

*) develop appn preventing SQL Injection problem.

28/11/2012

→ jdbc driver name, url, db username and DataBase Password to gather are called Jdbc properties. It is always recomended to pass these Jdbc properties to application from outside the application by taking the support of Properties file. When JDBC code is placed Struts Action class we can supply JDBC properties to that Actionclass by using Properties file supporting.

(*) Procedure to place and use Jdbc properties in the properties file of Struts application..

Step-①:- Keep NetBeans IDE based StrutsApp1 ready

Step-②:- add new properties file to project.

Right Click on Src packg folder → new → propertie sFile → Filename: myfile → finish.

Step-③:- make sure that multiple properties files are configured in Struts configuration file with logical names.

in Struts-config.xml

<message-resources parameter = "com/myapp/struts/ApplicationResource" key="f2"/>
(ApplicationResource ← default file given by IDE.)

<message-resources parameter = "myfile" key = "f1"/>

Step-④:- Add JDBC properties in properties file.

myfile.properties

#Jdbc Properties

my.driver = sun.jdbc.odbc.JdbcOdbcDriver

my.url = jdbc:odbc:oradsn

my.dbuser = scott

my.dbpwd = tiger

Step-⑤:- write following code in the execute(-,-,-,-) of RegisterAction class to read and use Jdbc properties from properties file

```
//read form data from FormBean class obj
RegisterForm fm = (RegisterForm) form;
String un = fm.getUsername();
```

```
String ps = fm.getPassword();
//Locate properties file  ---> protected method of predefined Action class.
MessageResources mr = getResources(request, "f1");
                                            ---> logical name of properties file
// read jdbc properties from the properties file
String s1 = mr.getMessage("my.driver");
String s2 = mr.getMessage("my.url");
String s3 = mr.getMessage("my.dbuser");
String s4 = mr.getMessage("my.dbpwd");
                          ---> key in the properties file.
//Write jdbc code
Class.forName(s1);
Connection con = DriverManager.getConnection(s2,s3,s4);
.....
```

*) We can Configure ActionServlet either with extention match or Directive match URL pattern in web.xml file In order to see it as Front Controller.

Configuring ActionServlet with directive ~~extention~~ match url pattern:-

in web.xml

```
<servlet>
    <s-n> action </s-n>
    <s-c> org.apache.struts.action.ActionServlet </s-c>
</servlet>
<servlet-mapping>
    <s-n> action </s-n>
    <u-p> /x/y/* </u-p>
</servlet-mapping>
```

→ Based on the Form Component type in Form page we need to design form bean properties in form bean class.

Form Component	Form Bean property
TextBox	String <variable> (Other data type is allowed)
Password Box	String <variable> (" ")
Radio Btn	String <variable> (" ")
Check Box (single)	String <variable> (" ")
Check Boxes (multiple Boxes with same name)	String[] <variable> (" ")
Combo Box (Select Box)	String <variable> (" ")
List Box	String[] <variable> (" ")
Submit Btn	String <variable> (" ")
File Upload	org.apache.struts.upload.FormFile(I)

(*) procedure to develop Struts application having all types of Form components in Form page. (Using NetBeans IDE).

Step-(1):- Create Web project having Struts capabilities

File → New project → Filename: StrutsApp2 → GlassFish 2.1 → ☑ struts-1.3.8

☑ Add tld files → finish.

Step (2):- Add the form page (input.jsp) in the WebPage folder of the project.

New → jsp: input → finish.

WebPages → web inf → tld files

input.jsp :-

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<html:form action="ipurl">
    <table border="1">
        <tr>
            <td> Name : </td>
            <td> <html:text property="name"/> </td>
        </tr>
```

```
<tr>
    <td> Age </td>
    <td> <html:password property="age" /> </td>
</tr>
<tr>
    <td> Address </td>
    <td>
        <html:textarea cols="20" rows="5" property="addrs">
            enter address
        </html:textarea>
    </td>
</tr>
<tr>
    <td> Gender : </td>
    <td>
        <html:radio property="gender" value="M"> Male </html:radio>
        <html:radio property="gender" value="F"> FeMale </html:radio>
    </td>
</tr>
<tr>
    <td> Marital Status </td>
    <td> <html:checkbox property="MS" value="Married"> Married </html:checkbox>
</tr>
<tr>
    <td> Qualification </td>
    <td>
        <html:select property="qlfy">
            <html:option value="engg"> Engineer </html:option>
            <html:option value="medico"> Medicine </html:option>
            <html:option value="arts"> Arts </html:option>
            <html:option value="commerce"> Commerce </html:option>
        </html:select>
    </td>
</tr>
<tr>
    <td> Cources </td>
    <td>
        <html:select multiple="yes" property="crs">
            <html:option value="java"> JavaPkg </html:option>
            <html:option value=".Net"> .NETPkg </html:option>
```

standalone attribute (select box is converted into list box)

```
<html:option value="oracle"> Oracle pkg </html:option>
</html:select>
</td>
</tr>

<tr>
<td> Hobbies </td>
<td>
<html:checkbox property="hb" value="stamp"/> Stamp collection
<html:checkbox property="hb" value="reading"/> Reading books
<html:checkbox property="hb" value="roaming"/> Travelling
</td>
</tr>
<tr>
<td> <html:submit value="send" /> </td>
<td> <html:reset value="reset"/> </td>
</tr>
</table>
</html:form>
```

NOTE:- It is always recomended to take the html form page content as html table content to align the form content quite easily as shown above.

29/11/2012

step-③:- add formbean class to the project.

Right click on src source package folder → New → Other → struts → Struts Action Form Bean → next → class name : InputForm → package : P1 → Finish

```
// inputForm.java
public class InputForm extends org.apache.struts.action.ActionForward
{
    private String name;
    private int age;
    private String addrs;
    private String gender;
    private String ms;
    private String qlfy;
    private String[] crs;
    private String[] hb;
```

// place getXxx() and setXxx() methods

}

select the above member variable → Right click → Insert Code → generate Setter & getters ....

Step-4 :- add Action Class to the project

Right click on project → new → other → Struts → Struts Action → next →

Class Name : InputAction → Package : P1 → Action Path : /ipath → next

→ Action Form Bean Name : InputForm → Finish

Step-5 :- Configure InputAction class in Struts configuration file as shown below.

in struts-config.xml

```
<action-mappings>
  <action name="InputForm" path="/ipath" scope="session" type="P1.InputAction">
    <forward name="res" path="/result.jsp" />
  </action>
</action-mappings>
```

Step-6 :- Write following code in the execute(-,-,-,-) of InputAction.java

```
public ActionForward execute(-,-,-,-) throws exception {
    //read Form data from FormBean class obj
    InputForm fm = (InputForm) form;
    String name = fm.getName();
    int age = fm.getAge();
    String gen = fm.getGender();
    String addrs = fm.getAddrs();
    String ms = fm.getMs();
    String qlfy = fm.getQlfy();
    String crs[] = fm.getCrs();
    String hb[] = fm.getHb();
    //write to Business logic
    if (age>=18)
        request.setAttribute("msg", "Elegible to vote");
    else
        request.setAttribute("msg", "Not Elegible to vote");
```

```
//keep Form data in request attributes
request.setAttribute("name", name);
request.setAttribute("age", age);
request.setAttribute("gender", gen);
request.setAttribute("addrs", addrs);
request.setAttribute("ms", ms);
request.setAttribute("qlfy", qlfy);
request.setAttribute("crs", crs);
request.setAttribute("hb", hb);

//returns ActionForward class obj
return mapping.findForward("res");
}//execute(-,-,-,-)
}//class
```

Step-7:- add the result.jsp to the webpages folder of the project.

Right click on webpages folder → new → Jsp → JspFilename : result

result.jsp

```
Name is : <%= request.getAttribute("name") %> <br>
Age is : <%= request.getAttribute("age") %> <br>
Address is : <%= request.getAttribute("addrs") %> <br>
Marital Status : <%= request.getAttribute("ms") %> <br>
Gender : <%= request.getAttribute("gender") %> <br>
Qualification : <%= request.getAttribute("qlfy") %> <br>
Courses : <% String crs[] = (String[]) request.getAttribute("crs");
            for(int i=0; i<crs.length; ++i)
            out.println(crs[i]+"....."); %> <br>
Hobbies : <% String hb[] = (String[]) request.getAttribute("hb");
            for(int i=0; i<crs.length; ++i)
            out.println(hb[i]+"....."); %> <br>
Result is : <%= request.getAttribute("msg") %>
```

Step-8 :- Run the project

Right click on project ⟶ Run as

request URL :- http://localhost:2020/StrutsApp1/input.html

→ Except List box, check box components the remaining Form components of Formpage use request parameters either with empty values (or) with default values even though they are not filled up (or) their items are not selected.

Improvise result.jsp of above application by using Struts supplied Jsp tags.

result.jsp

```
<%@taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
          prefix="bean" %>
          prefix="logic" %>

<logic:notEmpty name="name">
  Name is: <bean:write name="name" /> <br>
</logic:notEmpty>

-
-
-
-
-
-

<logic:notEmpty name="crs">
  Courses:
  <logic:iterate id="item" collection="<%=request.getAttribute("crs")%>">
    <bean:write name="item" />.....
  </logic:iterate>
</logic:notEmpty>
<br>

<logic:notEmpty name="hb">
  Hobies
  <logic:iterate id="item" collection="<%=request.getAttribute("hb")%>">
    <bean:write name="item" />.....
```

Acts as for loop

in java code we using for loop for iteration but we using jsp taglibs in struts we use logic:iterate

→ counter variable pointing to each element of given array in each iteration.

→ gives string array

→ data structure

```
</logic:iterate>
</logic:notEmpty>
<br>
```

Result is : <bean:write name="msg" />

Understanding FormBean life cycle :-

Browser window launches the Formpage

↓

ActionServlet creates/locates the form page related form bean class

↓

if created, the FormBean class object will be placed in specified scope

↓

ActionServlet calls getXxx(-) on FormBean class obj and reads data from FormBean properties and also writes that data to Form components as initial values

↓

ActionServlet traps the Formpage generated request

↓

ActionServlet creates/locates FormBean class object
(if created, The FormBean class object will be placed in specified scope)

↓

ActionServlet calls reset(-,-) on FormBean class obj.

↓

ActionServlet calls setXxx(-) on FormBean class obj to write the received Form data of Form page to FormBean Properties.

↓

ActionServlet calls Validate(-,-) method on FormBean class object to Validate form data.

if validations Errors are there

ActionServlet passes the control to input page to display formValidation error msgs

if Validation errors are not there

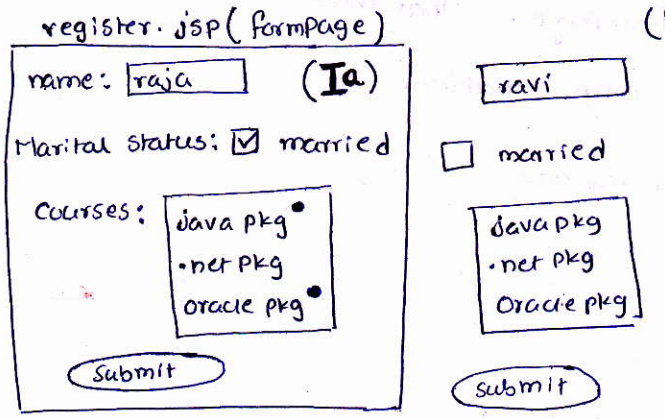
ActionServlet creates/locates Action class obj and calls execute(-,-,-,-) method on it.

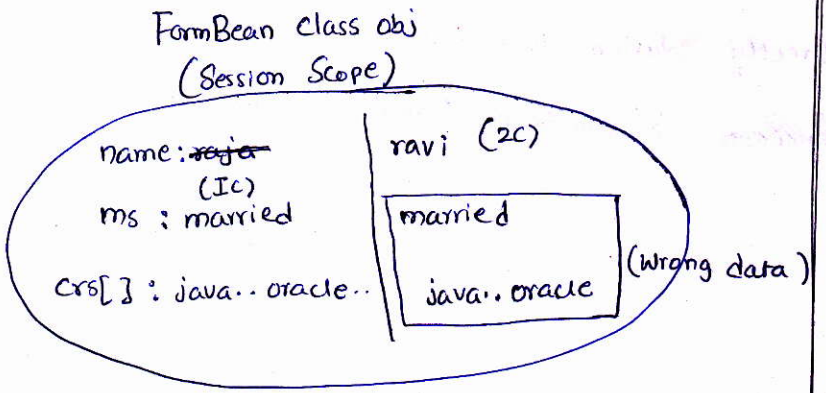
30/11/2012

→ List box, check box component Send their parameter value to Server only when their items are selected, otherwise no request parameter will be formed related to these components.

Understanding the problem related to checkbox, listboxes while working with Session Scoped FormBean :-

→ FormBean properties related to checkbox, listbox holds wrong data while working with Session Scoped FormBean. that means they hold selected state eventhough they are not selected by enduser in form page. This problem raises when same form page is submitted multiple times from same browser window.





RegisterForm.java (FormBean class)

```
Public class RegisterForm extends ActionFor
{
   //FormBean properties
    String name;
    String ms;
    String crs[];
   public void setXxx(-)
    {
      ----- (1b) → 3 methods
      ----- (2b) → 1 method
                    (only setName(-)
    }
   Public String|String[] getXxx()
    {
      ----
    }
}
```

1a, 1b, 1c → represents 1st request flow

2a, 2b, 2c → represents 2nd request flow

note: Assume that both requests are given from same browser window.

There are two approaches to solve the above problem.

Solution 1 / Approach 1 :-

→ change the FormBean scope to "request" scope

When FormBean scope is "request" ActionServlet creates new FormBean class object for every request related to FormPage. That means FormBean properties can't hold previous request data in any situation.

in struts-config.xml

change scop="request"

FormBean class obj for request1

name : raja
ms : married
crs[] : java.. oracle

FormBean class obj for request2

name : ravi
ms : null
crs[] : null

Note:- This Solution 1 is not directly solution to resolve the above problem. To solve the above problem having "Session" scope FormBean takes the support of reset(-,-) method.

Approach ② / solution ② :- (Working with reset(-,-) in Session scoped FormBean).

→ reset(-,-) placed in FormBean class executes for every request before executing setXxx(-) methods. This method is originally available in predefined ActionForm class and its default implementation does nothing.

→ when we override this method in our FormBean class we set not selected state default values to the FormBean properties that are representing checkboxes, listbox components in Session scoped FormBean.

→ There is no need of overriding this method when the FormBean scope is "request"

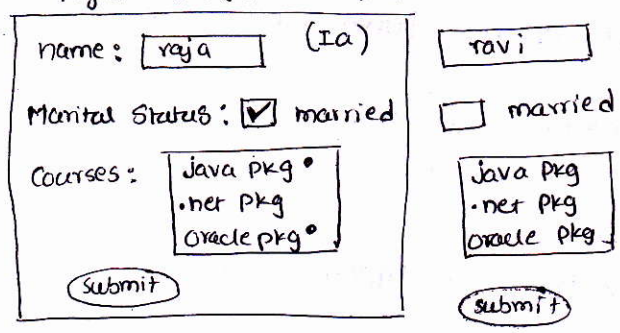
→ There are two Forms of reset(-,-)

1) public void reset(ActionMapping mapping, HttpServletRequest req)

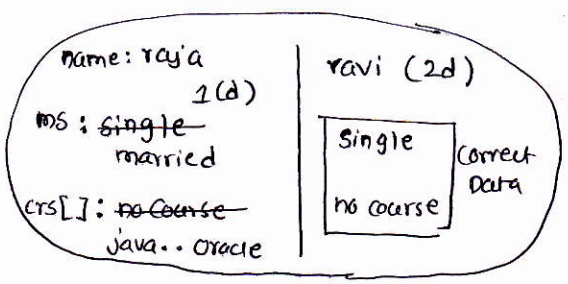
2) public void reset(ActionMapping mapping, ServletRequest req)

1st form is recomended to use.

register.jsp (form page)



FormBean class Obj (Session scope)



RegisterForm.java (FormBean class)

```
public class RegisterForm extends ActionForm
{
    // FormBean properties
    String name;
    String ms;
    String crs[];

    public void reset(-,-)
    {
        ms = "single";        (1b)
                              (2b)
        crs[] = new String[1]; crs[0] = "no course"
    }

    public void setXxx(-)
    {
        ---    (1c) ⟶ 3 methods
        ---    (2c) ⟶ 1 method
    }                 (only setName(-))

    public void getXxx()
    {
        ---
        ---
    }
}
```

note: Assume that both requests are given in same browser window!

→ while placing logics in reset(-,-), we want to know much details about current Action class & current request then use the parameters of reset(-,-) method. and we can use mapping object to gather much details about current Action class. Similery we can use request Object to gather more details about current request.

01/12/2012

Form Validations :-

Veryfying the format and pattern of the Form data is called FormValidation, is always recomended to use validated Form data as input values of business logics, persistence logics.

Eg:- checking weather required fields are typed or not

weather email id is having ., @ symbols or not

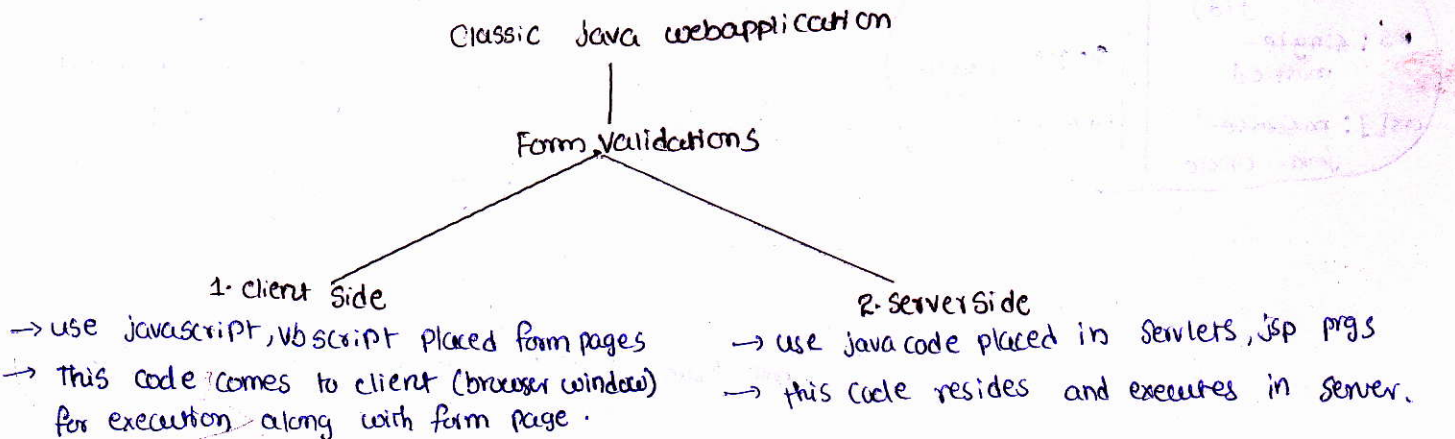
Q:- What is the difference blw FormValidation logic and Business logic?

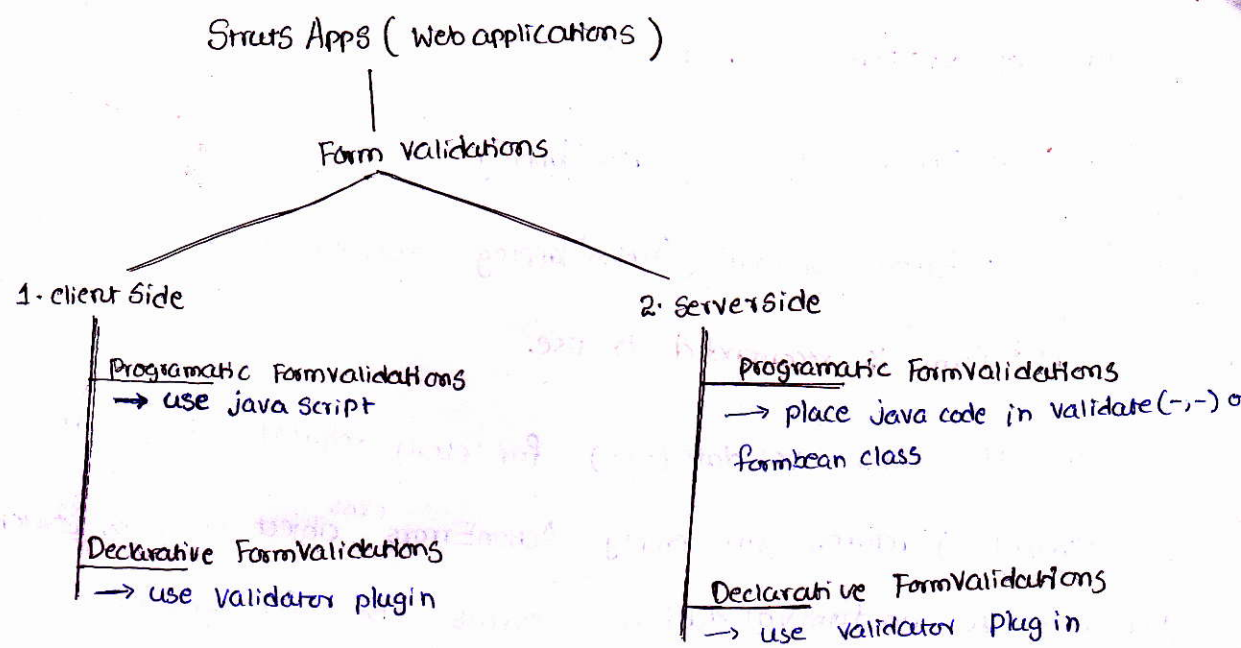
→ In FormValidation logic the format of the Form data will be verified.

→ In Business logic results will be generated by taking Formdata as Input data.

Eg:- checking weather username, passwords are typed or not is called Form validation.

checking weather given username, passwords are there in DataBase s/w or not is Business logic/ persistance logic.





→ Validator plugin supplies 14+ built-in form validation logics | rules having both java code, java script code.

→ programer developed explicit Form Validation logics are called programatic logics.

→ working with Validator plugin supplied readymade FormValidation logics is nothing but declarative Form Validations.

→ For better FormValidations use both programatic and Declarative Formvalidation logics together.

→ only client side FormValidations reduces network round trips between client side and Server side. but there is the chance of disabling Script code execution two browser Settings.

→ only Server Side FormValidations increases network round trips b/w client side and Server side.

*** V.Imp.

→ place both client side and Server side FormValidation logics but make sure that Server side FormValidation logic is executing only when client side Form validations are not done. This is industries best practice to use.

→ To perform server side Form validations placed the Java Stmts based FormValidation logic in the Overridden validate method of our FormBean class.

Prototype of validate(-,-) method :-

1) public ActionErrors validate(ActionMapping mapping, ServletRequest req)

2) public ActionErrors validate(ActionMapping mapping, HttpServletRequest req)

2nd form is recomended to use.

→ ActionServlet calls validate(-,-) for every request as FormBean lifecycle method.

→ If validate(-,-) returns an empty ActionErrors class object to ActionServlet indicating that there are no form validation errors then ActionServlet transfers the control to execute(-,-,-,-) method of ActionClass to execute the Business class.

→ If validate(-,-) return ActionErrors class object with elements indicating that there are FormValidation errors then ActionServlet transfers the control to input Jsp page.

→ Jsp that is having logic to display FormValidation errors is called input page. It can be ~~Every~~ any Jsp program including FormPage

Ⓐ Procedure to apply Server side programatic FormValidations (required rule) in Struts application.

Step-I:- Keep the 1st struts application ready.

Step-II:- Override the validate(-,-) in FormBean class as shocon below.

in RegisterForm.java

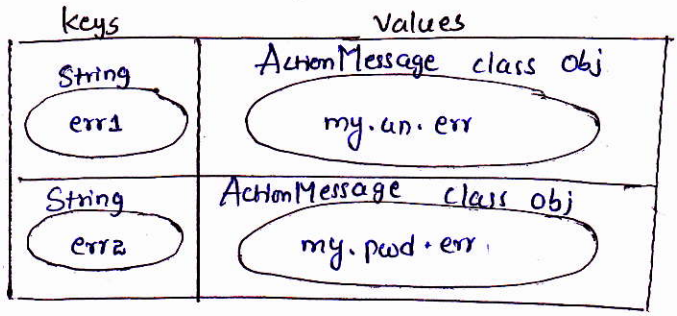
```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest req)
{
    S.o.p("RegisterForm : validate(-,-)");
    ActionErrors errs = new ActionErrors();
```

```
// write server side programatic form validation logic (required rule) (java code)

if(username==null || username.length()==0 || username.equals(""))
  {
    errs.add("err1", new ActionMessage("my.un.err"));
  }
if(password==null || password.length()==0 || password.equals(""))
  {
    errs.add("err2", new ActionMessage("my.pwd.err"));
  }
S.O.P("errs obj size" + errssize());
return errs;
} //validate(--)
} //class
```

"err1" → logical name of validation error

"my.un.err" → key in the properties file.

ActionErrors class object



ActionErrors class object represents all FormValidation errors. In that ActionMessage class object represent FormValidation error message of each validation error. These messages can be gathered from properties file by specifying "keys".

Step-III :- add the FormValidation error messages in properties file.

in myFile.properties

```
#form validation error msgs
my.un.err = username is required
my.pwd.err = password is required
```

→ Form Validation error messages.

Step-IV :- make sure that the above properties file is configured in Struts Configuration file.

in Struts-config.xml

```
</action-mappings>
<message-resources parameter="myfile" />
</struts-config>
```

05/12/2012

→ In ordered to add client side programatic form validation to Struts Application placed java script in form page (Register.jsp) as shown below.

Procedure to have both client-side and server-side programatic FormValidation logic in Struts application. but make sure that Server side FormValidations logics are executing only when client-side Form Validations are not done.

Step-I :- keep the previous ~~Struts-pv~~ Strutsapp-pv application ready (refer previous day class)

Step-II :- Write following javascript code in Formpage and call it against onsubmit event.

Register.jsp

```
<%@ taglibs %>
<html:html>
  <head>
    <script language="JavaScript">
     function myValidate(frm)
     {
        //read form data
     Var un = frm.username.value;
     Var pwd = frm.password.value;

//write client side programatic form validation logic
     if(un == "") //required rule
      {
        alert("user name is manditory");
        frm.username.focus();
        return false;
      }
     else //first char must be an alphabet
      {
       Var val = un.charCodeAt(0); //gives ascii value of first char
if(!(val>=65 && val<=90) && !(val>=97 && val<=122))
  {
    alert("First char must be an alphabet");
     frm.username.focus();
     return false;
  }//if
}//else
  if(pwd == "") //required rule
  {
```

```
        alert("password is manditory");
        frm.password.focus();
        return false;
    }
frm.vflag.value = "true"; //to indicate that Client side Validations are done.
                          ↳ to maintain a flag holding the status of client-side Form Validation logic execution.
    return true;
}//myvalidate(-)
</script>
</head>
```

```
<center> <h1> <bean:message key="my.title" /> </h1> <center>
<html:form action="register" onsubmit="return myValidate(this)">
                                          *1
```

username, password text boxes

submit button

```
<html:hidden property="vflag" value="false" />
</html:form>
```

logic to display results.

(*1) calling the Javascript function against "onsubmit" event (raises when submit button is clicked. In this statement "return" is manditory)

"this" return keyword returns true/false given by myValidate(-) function call to browser window. and Browser window takes the decision of blocking request going to Server or Sending request to server based on this Value. (true/false).

↳ when return value is false

↳ when return value is true.

Step-3: Write following code in the Validate(-,-) of FormBean class in RegisterForm.java

```
Public ActionErrors validate(-,-)
{
    S.O.P("RegisterForm: validate(-,-)");
    ActionErrors errs = new ActionErrors();
    //read vflag value
String flag = req.getParameter("vflag"); //gives client side form Validation logic execution status
```

05-12-2012

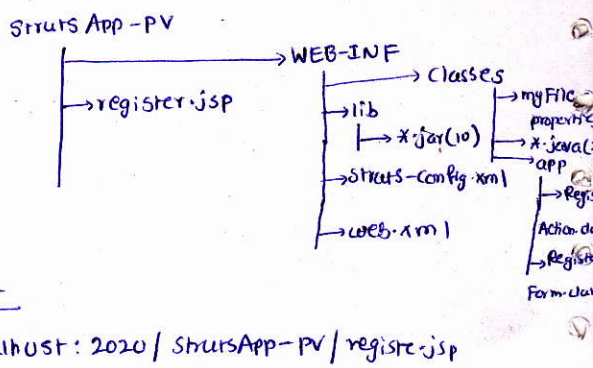
```
>>>>>>>>>>>> Struts App with Server Side Programatic Form Validations >>>>>>>>>>>>>>>>
----------register.jsp----------
<%@taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<html:html>
  <head>
      <script language="JavaScript">
          function myValidate(frm)
            {
                //read form data
                var  un=frm.username.value;
                var pwd=frm.password.value;

                //write client side programatic  form  validation logic
                if(un=="")  // required rule
                {
                     alert("user name is mandatory");
                     frm.username.focus();
                     return false;
                }
                else  // first char must be an alphabet
                {
                     var val=un.charCodeAt(0); //gives asci value of first char
                     if( !(val>=65 && val<=90) && !(val>=97 && val<=122))
                     {
                          alert("First char must be an alphabet");
                          frm.username.focus();
                          return false;
                     }//if
                }//else
                if(pwd=="") // requied rule
                {
                     alert("password is mandatory");
                     frm.password.focus();
                     return false;
                }
                frm.vflag.value="true";  // to indicate that Client side validations are done
                return true;
            }//myvalidate(-)
        </script>
    </head>
 <center> <h1> <bean:message key="my.title" /></h1> </center>
<html:form action = "register" onsubmit="return myValidate(this)">
  <bean:message key="my.user"/><html:text property = "username"/>
        <html:errors property="err1"/> <br>
  <bean:message key="my.pwd"/><html:password property = "password"/>
        <html:errors property="err2"/> <br>
     <html:submit>
     <bean:message key="my.btn.cap"/>
     </html:submit>
     <html:hidden property="vflag" value="false"/>
</html:form>

 <logic:notEmpty name="msg" >
     <bean:message key="my.res"/> <bean:write name="msg" />
 </logic:notEmpty>

  <br>
   <bean:message key="my.footer"/>
</html:html>
----------web.xml----------
<web-app>
 <servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
   <param-name>config</param-name>
   <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

Client-side programatic form Validation logic

```
73
74    <servlet-mapping>
75      <servlet-name>action</servlet-name>
76      <url-pattern>*.do</url-pattern>
77    </servlet-mapping>
78  </web-app>
79  ----------struts-config.xml----------
80  <!DOCTYPE struts-config PUBLIC
81          "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
82          "http://struts.apache.org/dtds/struts-config_1_3.dtd">
83
84  <struts-config>
85
86     <form-beans>
87           <form-bean name="rf" type="app.RegisterForm"/>
88     </form-beans>
89
90     <action-mappings>
91        <action   path="/register" type="app.RegisterAction" name="rf" input="/register.jsp">
92                <forward name="result" path="/register.jsp"/>
93              </action>
94     </action-mappings>
95        <message-resources parameter="myfile"/>
96  </struts-config>
97  ----------myfile.properties----------
98  # presentation lables
99  my.title=<center><h1> Login Page </h1></center>
100 my.user= UserName
101 my.pwd= Password
102 my.btn.cap=Login
103 my.res=Result is
104 my.footer=@copy; copy rights reserved 2011-12
105
106 #form validation error msgs
107 my.un.err=username is required
108 my.un.fchar=user name must begin with an alphabet
109 my.pwd.err=password is required
110 # To apply styles on form validation error msgs
111 errors.header=<font color=red size=1>
112 errors.footer=</font>
113 ----------RegisterForm.java----------
114 package app;
115 import org.apache.struts.action.*;
116 import javax.servlet.http.*;
117
118 public class RegisterForm extends ActionForm
119 {
120      private String username="raja";
121      private String password="hyd";
122
123      public RegisterForm()
124      {
125           System.out.println("RegisterForm: 0-param constructor");
126      }
127
128      public void setUsername(String username)
129      {
130           System.out.println("RegisterForm:setUsername(-)");
131           this.username = username;
132      }
133      public String getUsername()
134      {
135           System.out.println("RegisterForm:getUsername()");
136           return username;
137      }
138      public void setPassword(String password)
139      {
140           System.out.println("RegisterForm:setPassword(-)");
141           this.password = password;
142      }
143      public String getPassword()
144      {
```

```
145     System.out.println("RegisterForm:setPassword(-)");
146           return password;
147      }
148
149  public ActionErrors validate(ActionMapping mapping,HttpServletRequest req)
150  {
151
152      System.out.println("RegisterForm:validate(-,-)");
153      ActionErrors errs=new ActionErrors();
154      // read vflag value
155      String flag=req.getParameter("vflag"); // gives Client form validation logic execution status
156      if(flag.equals("false"))
157      {
158         System.out.println("server side form validation is going on.....");
159         // write server side programatic form validation logic(required rule) (java code)
160        if(username==null || username.length()==0 || username.equals(""))
161         {
162         errs.add("err1",new ActionMessage("my.un.err"));
163         }
164        else   // to check weather first char is alphabet or not
165        {
166           // get First char  from username
167           char fchar=username.charAt(0);
168           if(!Character.isUpperCase(fchar) && !Character.isLowerCase(fchar))
169           {
170            errs.add("err1",new ActionMessage("my.un.fchar"));
171           }//if
172        }//else
173        if(password==null || password.length()==0 || password.equals(""))
174        {
175           errs.add("err2",new ActionMessage("my.pwd.err"));
176        }
177       System.out.println("errs obj size"+errs.size());
178      }//if
179      return errs;
180   }//validate(-,-)
181  }//class
182  ------------------------------RegisterAction.java----------------------------
183  package app;
184  import org.apache.struts.action.*;
185  import javax.servlet.http.*;
186  import javax.servlet.*;
187
188  public class RegisterAction extends Action
189  {
190
191      public RegisterAction()
192      {
193          System.out.println("RegisterAction: 0-param constructor");
194      }
195
196      public ActionForward execute(ActionMapping mapping,ActionForm form,HttpServletRequest req,HttpServletResponse res
197      {
198          System.out.println("RegisterAction:execute(-,-,-,-)");
199          RegisterForm rf = (RegisterForm)form;
200          String user = rf.getUsername();
201          String pass = rf.getPassword();
202
203          if(user.equals("Sathya") && pass.equals("tech"))
204          {
205              req.setAttribute("msg","Valid Credentials");
206          }
207          else
208          {
209              req.setAttribute("msg","InValid Credentials");
210          }
211
212          return mapping.findForward("result");
213      }
214  }
215
```

SeverSide Programmatic FormValidation logic



request url's

http://localhost:2020/strutsApp-pv/register.jsp

```
if( flag.equals("false") )
{
    s.o.p("serverside formvalidation is going on.....");
    ..... } sever side FormValidation logic using Java code.
}
return errs;
}
```

NOTE: Deployement directory structure is same as previous application (StrutsApp-PV)

Procedure to disable Script code execution in Internet Explorer

Tools menu → Internet options → security (tab) → Custom level → scripting → Active Scripting → ⊙ Disable

→ For above steps based Struts application if both client-side and server-side form validation logic refer supplimentary handout given on 05-12-2012

→ to hide JavaScript code visibility from end users place JavaScript code in seperate ".js" file and link that file with formpage as shown below.

Validator.js (in StrutsApp-PV folder)

```
function myValidate(frm)
{
    ..... // same as previous appn
}
```

in register.jsp

```
.....
<html:html>
  <head>
    <noscript> please Enable JavaScript </noscript>
    <script language="JavaScript" src="Validator.js">
    </script>
  </head>
  ..... // same as previous appn
</html:html>
```

06/12/2012

→ Struts 1.x slw supplies two built-in plugins to simplify the activities of programmer

① Validator plugin ⟶ given to perform client side and server side Declarative Form Validations

② Tiles plugin ⟶ given to develop layout based webpages in struts appn.

→ Every struts plugin is identified with its plugin class that implementes org.apache.struts.action.PlugIn (I) and we must configure this plugin class in struts cfg file using <plug-in> tag.

→ To cfg Validator plugin in struts cfg file specify its plugin class (Org.apache.struts.validator.ValidatorPlugIn) as shown below

```
<plug-in  className = "org.apache.struts.validator.ValidatorPlugIn">
    <set-property  property = "pathnames"  value = "/WEB-INF/validator-rules.xml,
                                                    /WEB-INF/validation.xml" />
</plug-in>
```

→ Validator plugin can not be used outside the struts 1.x appns.

→ Validator plugin gives 14+ predefined Validator rules having java code, java script code to perform server side, client side formValidations respectively.

→ these Validator rules are

required
minlength
maxlength
date
intRange
byte
int
email
and etc.....

(Client side)

→ the .js files of commons-validator-1.3.1 represents the formValidation logics of these 14+ validator rules.

→ validateXxx(-) of Org.apache.struts.validator.FieldChecks (C) contains formValidation logics (server side) of these 14+ Validator rules

→ To use Validator plugin in Struts application the FormBean class must extended from org.apache.struts.validator.ValidatorForm (c) (It is subclass of org.apache.struts.action.ActionForm (c))

Q:- When Validator plugin will be recognized and will be activated in struts application

Ans:- When ActionServlet reads Struts Configuration file then it recognizes Validator plugin. This generally happens either during Server Startup or during the deployment of Struts application.

When ActionServlet calls Validate(-,-) of predefined ValidatorForm (c) (super class of our FormBean class) the Validator plugin will be activated and will be used for FormValidations.

NOTE:- The Validate(-,-) of ActionForm (c) does not contain logic to activate ValidatorPlugIn where as the Validate(-,-) of ValidatorForm (c) contains this logic. so we must make our form bean class extending from ValidatorForm (c) to use ValidatorPlugin in our Struts app.

→ Validate(-,-) of our FormBean class performs serverside programatic FormValidations. where as the Validate(-,-) of our FormBean superclass contains logic to perform the ValidatorPlugin based declarative FormValidations by using the entries of Validator-rules.xml file, Validation.xml file.

→ Validator-rules.xml Contains the configuration details of predefined Validation rules belonging to validator plugin. like Validator rule name, DefaultErrormessageKey, .js filename function name, class and method name where FormValidation logics are available.

→ This File is useful only while adeling Custom Validator rules to Validator plugin.

→ Validation.xml is the imporent resource while working with ValidatorPlugIn in this file. we map Validator rules with FormBean properties, we supply argument values required for logics and default error msgs.

→ The default error msgs of Validator rules are given with arguments {n}. so, that they can be used for multiple Form components by supplying appropriate values to arguments

Example:- The Default error msg of required rule

Errors.required = {0} is required

When required rule is applied on username

The error msg is : username is required
{0}

when required rule is applied on password

The error msg is : password is required
{0}

Note:- {0} will be supplied from Validation.xml file.

→ we can gather default error msgs of validator rules from the comments of validator-rules.xml file

→ we can get validator-rules.xml file by extracting Struts-core-1.3.8 jar file.

→ procedure to use validator plugin to perform serverside Declarative FormValidations on StrutsAppns (required rule)

Step ①:- keep 1st Struts application ready

Step ②:- Configure validator plugin supplied plugin class in Struts cfg file as the last subtag of <struts-config> tag 58-72

in struts-cfg.xml

```
<plug-in  className = "org.apache.struts.validator.ValidatorPlugIn">
  <set-property  property = "pathnames"  value = "/WEB-INF/validator-rules.xml,
                                                  /WEB-INF/validation.xml" />
</plug-in>
```

NOTE:- we can gather above code from struts-cfg.xml file of <struts_home>/apps/struts-blank-1.3.8.war file extraction

Step-③:- place validator-rules.xml file in WEB-INF folder

NOTE:- Search in Struts s/w installation

Step-④:- Create properties file having default error messages 204-219

myfile.properties (WEB-INF/classes folder)

```
# Struts Validator Default error messages
errors.required = {0} is required
errors.minlength = {0}
- - - - -
- - - - - -
- - - - - - -
```

Gathers these msgs from the comments of validator-rules.xml

Step-⑤:- Configure the above properties file in Struts configuration file after </action-mappings> (56)

```
<message-resources  parameter = "myfile" />
<plug-in  - - - - />
```

Step-⑥:- make sure that FormBean class is extending from ValidatorForm class and it should not containing validate(-,-) (94)

RegisterForm.java

```
Package app;
  import org.apache.struts.validator.*;
public class RegisterForm extends ValidatorForm
  {
    ----
    ----
    ----
    ----
    ----
  }
```

Javac -d . RegisterForm.java

Step-⑦ :- add validation.xml in WEB-INF folder to specifying validator rules on FormBean properties. (64)

in validation.xml

```
<!DOCTYPE ------ Validator_1-3-0.dtd>
<form-validation>
  <formset>
    <form name="rf">                    → FormBean logical name
      <field property="username" depends="required">   → FormBean property name
                                                       → Validator rule
        <arg0 key="my.un"/>
                  → This key related msg of properties file supplies {0} value of required rule error msgs.
      </field>
      <field property="password" depends="required">
        <arg0 key="my.pass"/>
      </field>
    </form>
  </formset>
</form-validation>
```

Step-⑧ :- add user defined messages in properties file to supply argument values to Default error messages (221-222)

in myfile.properties

```
# User-Defined msgs supplying arg values to Default error msgs
  my.un = Login username
  my.pass = Login password
```

→ supplies {0} values of the required rule error msgs.

Step-⑨:- configure input page for Action class (51)

in struts-cfg.xml

```
<action-mappings>
    <action path="/register"  type="app.RegisterAction"  name="rf"
                                                  input="/register.jsp">
        =
        =
        <forward name=
    </action>
</action-mappings>
```

Step-⑩:- add <html:errors/> in the input page (12)-(14)

in register.jsp

```
<%@ taglib ---
---
<html:errors/>
: : :
```

Step-⑪:- Test the applications.

NOTE:- while working with validator plug-in the logical names of generated validation errors are same as FormBean properties.

So we can use FormBean property names in property attribute of <html:errors> tag to display Validation error msgs with respect to the positions of FormComponents.

in Register.jsp

```
<html:form  action="register">
Login username : <html:text  property="username"/>  <html:errors property="username"/>
(*1)
Login password : <html:password  property="password"/> <html:errors property="password"/>
(*1)
```

(*1) here username, passwords are Formbean propertynames and also acting as the logicalnames of formvalidation errors

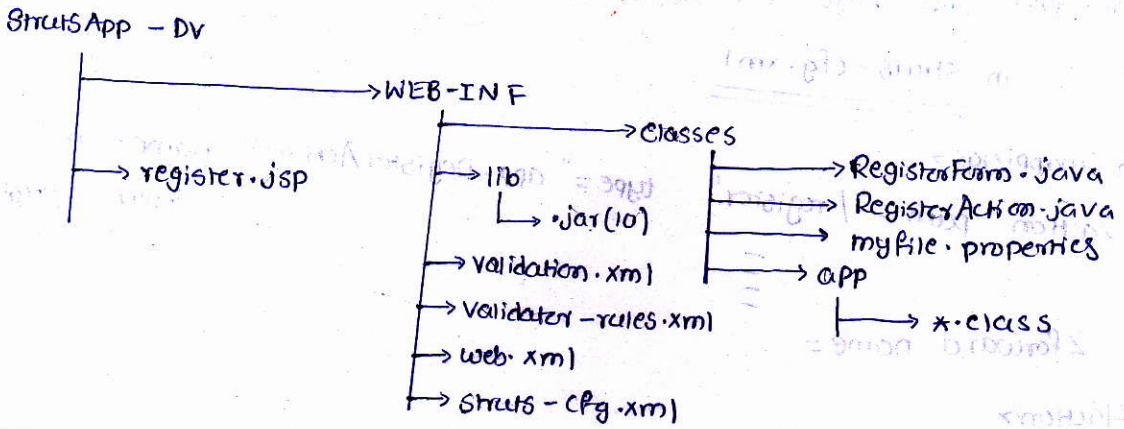
** For the above steps based complete example is reffered page 18-22 (App②)

132 — 150 →

228 — 233 → delete

223 → delete

Deployment Directory Structure :



Request url :

http://localhost:2020/StrutsApp-DV/register.jsp

Flow of Execution While working with Validator plugin

(a) ActionServlet calls validate(-,-) on our FormBean class object, since not available the Superclass validate(-,-) will execute

(b) Superclass validator(-,-) activates validatorplugin and performs formvalidations on FormBeans properties data based on the configurations done in validation.xml file. In this process validator-rules.xml file will also be used.

(c) This Superclass validator(-,-) returns ActionErrors class object to ActionServlet,

(d) If ActionErrors class object is empty ActionServlet passes the control to Action class Otherwise passes the control to input page

Procedure to use Validator plug-in for Client-side Form Validations :

Step-I : Perform all the activities that are required for Server-side Form Validations.

keep StrutsApp-DV application is Ready.

Step-II : Add following additional lines in Form page

in register.jsp

```
<%@ taglib uri =
        ---------
        ---------
        ---------
<html:form action="register" onsubmit="return validateRf(this)">
<html:javascript formname="rf" />
        ----
        ---
```

(FormBean logical name; validateRf — fixed word)

NOTE:- <html:javascript> makes validator plug-in to generate javascript code related to validator rules in Formpage.

This code contains one-Dynamic javascript function having name validate form Bean logical name of (form) like validateRf(form)

formBean logical name

Q:- while working with validator plug-in how can we make server-side FormValidations logics are executing only when client-side formvalidations are not done!

Ans:- Send one additional flag to Server-side, indicating weather client-side FormValidations are done (or) not done and allows ActionServlet to call Superclass validate(-,-) based on that flag.

↳ Validator form

Step-I :- keep StrutsApp-DV ready

Step-II :- add Following code in Formpage to set flag indicating weather FormValidations are done (or) not

in register.jsp

```
<html>
  <head>
    <script language="javascript" />
        function myvalidate(frm)
        {
    var flag = validateRf(frm);
    frm.vflag.value = "yes";
     return flag;
        }
    </script>
  </head>
<html:form action="register" onSubmit="return myvalidate(this)">
    <html:javascript formName="rf" />
      :
      :
```

Step-III : add validate(-,-) on FormBean class and call super.validate(-,-) based on "flag". that indicates client-side FormValidation execution stmts.

in RegisterForm.java

```
public ActionErrors validate(ActionMapping mapping, HSR req)
    {       //read flag value
    String vstatus = req.getParameter("vflag");
            S.O.P(vstatus);
    ActionErrors err = new ActionErrors();
     if(vstatus.equals("no")) //when client-side validations are not done
      {
      //performs server side validator plug-in based FormValidations
         err = super.validate(mapping, req);
      }
      return err;
    }
}
```

→ While working with Validator plug-in validator rules, the {0} values of error msg should be passed from properties file and other arguments {1} ------ should be passed from Validation.xml file

Because {0} → is required to just-complete error msg

{1} {2} → are required to supply inputs to Validator rule Validation logic to complete the error msg.

Eg:- "minlength" validator rule error msg is errors.minlength = {0} can not be less than {1} characters

The complete errormsg is: Username can not be less than 5 characters

here {0} is just required to complete error msg.

But {0} is required to complete errormsgs and also as the input value of the error msg.

So pass {0} - value From properties - file and

{1} - value From Validation.xml file

→ Applying required, minlength rules on "username" property in Validation.xml

```
<field property="username" depends="required, minlength">
    <arg0 key="my.un" />
    <arg1 name="minlength" key="${var:minlength}" resource="false">
        <var>
            <var-name>minlength</var-name>
            <var-value>5</var-value>
        </var>
    </field>
```

Supplies {1} value at minlength rule

name = "minlength" — Validator rule name

key = "${var:minlength}" — indicates that is argument value should not be looked on properties file (resource bundle);

Supplies minlength variable value & that will be used as {1} value

→ While working with minlength, maxlength Validator rules the variable name that should be used to pass {1} - value must match with rule-name.

→ When multiple validator rules are applied on FormBean property and if we are looked to pass same value for any {n} argument of these rules related error msgs then use <arg key="n"> without specifying

any rulename (Refr above) that is arg0. otherwise we need to specify rule name in arg<n> refer the below validation.xml file <arg0>

Applying required, maxlength validator rules on password FormBean property.

In validation.xml of StrutsApp - Dv appn

```
<field property="password" depends="required, maxlength">
    <arg0 name="required" key="my.pass" />  ⟶ supplies {0} value of required rule msg error
    <arg0 name="maxlength" key="my.pass1" /> ⟶ supplies {0} value of maxlength rule msg error
    <arg1 name="maxlength" key="${var:maxlength}" resource="false" />
        <var>
            <var-name> maxlength </var-name>
            <var-value> 15 </var-value>
        </var>
</field>
</form>
```

Supplies {1} value of maxlength validator rule error msg.

*) Since the 14-Rules of Validator plugin are not at all sufficient for Realworld formvalidations it is always recommended to use them along with programmatic formValidation logics

That means use both programatic & Declarative approaches together in realtime scenario from FormValidations

There are three aproaches for this

① Using mask Validator rule : <Approach-1>

② Calling super.validate(-,-) from validate(-,-) of our FormBean class calling Validator plugin generated javascript function from core javascript function of formpage : <Approach-2>

③ Adding custom Validator rule to validator plug-in : <Approach-3>

Approach 1: mask validator

→ It is a predefined validator rule that allows us to keep custom validation logics in the form of RegulerExpression.

→ It can generated both Javacode, javascript code dynamically based on given reguler expression

→ The Default error msg of this Rule is : errors.invalied = {0} is invalied

→ we can use <msg> tag of validation.xml file to configure Custom error message for the validator rule of validator plugin we generally use this while working with mask-validator rule. Because "{0} is invalied" is incomplete error message.

*) applying required, mask rule on username property:

in validation.xml

```
<field property="username" depends="required,mask">
    <arg0 key="my.un" /> → supplies error message for mask rule
    <msg name="mask" key="my.mask.msg" />
    <var>
        <var-name> mask </var-name>
        <var-value> ^[0-9a-zA-Z]*$ </var-value>
    </var>
```

Reguler expression based form validation logic for mask-rule

↳ Reguler Expression allows only alpha numerics.

in myfile.properties

my.un = Login username

my.msk.msg = Username must have only alpha numerics.

→ RegulerExpression that allows only alphabets (or) only digits.

```
<var>
  <var-name> mask </var-name>
  <var-value> ^[a-zA-z] *$|^[0-9] *$ </val-value>
</var>
```

RegulerExpression for to allow minimum of 5 and max of 15 alphabets

```
<var>
  <var-name> mask </var-name>
  <var-value> ^[a-zA-z] {5,15} $ </var-value>
</var>
```

RegulerExpression to allow one or more characters except alphabets

```
<var>
  <var-name> mask </var-name>
  <var-value> ^[^a-zA-z] + $ </var-value>
</var>
```

(^ → negation)

RegulerExpression that allows zero or more alphabets in the following notation

```
<var>
  <var-name> mask </var-name>
  <var-value> ^([A-Z] [a-z])* $ </var-value>
</var>
```

NOTE:
For more other Syntatical reference for more Syntax details about RegulerExpression

regexp.txt of DVD

Limitations with mask rule :-

① We can't write validation logic by utilizing multiple formbean properties data. (checking weather password and retype password matching or not kind of logics is not possible with mask rule).

② Its validation logic is specific to one FormBean property.

③ mask rule related JavaScript is not executing for password kind of FormComponents

To overcome these problems use approach ②.

Approach 2 :-

→ This approach 2 is very good to write FormValidation logic based on multiple FormBean properties data. and also allows to reuse FormValidation logic within one formpage / FormBean class.

Example Application :-

Step-I :- keep StrutsApp-DV application ready (refer: 8th Date)

Step-II :- place the following FormValidation logic in the validate() of FormBean class.

in RegisterForm.java

```
public ActionErrors validate(ActionMapping mapping, HttpServletRequest req)
{
    //read vflag value
   String vstatus = req.getParameter("vflag");

ActionErrors errs = new ActionErrors();
if( vstatus.equals("no")) //when client side formvalidations are not done.
{
   //Performs Server side Validator plugin based form validations
    errs = super.validate(mapping, req);
//Write Server side form validation logic (Programmatic)
//to check weather username and password is having same length or not.
    if(errs.size() == 0)
    {
      if(username.length() != password.length())
       {
         errs.add("password", new ActionMessage("my.unpwd.msg"));
       } //if
    } //if
```

```
    } //if
    return errs;
  } //validate(-,-)
```

Step-III :- make sure that following validation logic is there in myValidate(-,-) of register.jsp.

```
<script language="JavaScript">
  function myValidate(frm)
  {
    var flag = validateRf(frm); //validator plugin generated function
    frm.vflag.value = "yes"; //indicates client side validations are
    if(flag == true)
    {
      //write client side form validation logic (programmatic)
      //to check weather username and password is having same length (or) not
  if(frm.username.value.length != frm.password.value.length)
   {
   alert("username and password must have same length");
     frm.username.focus();
     flag = false;
   } //if
  } //if
    return flag;
  } //myvalidate(-)
</script>
</head>
```

→ make sure that the following msg is added to properties file in myfile.properties

```
my.unpwd.msg = username and password must have same length.
```

12-12-12

Approach 3 (add custom validator rule to Validator plugin) :-

NOTE:- The custom rule added to Validator plugin can be used the multiple FormBean classes of one struts appn. It always recomended to develop this custom rule based on predefined validator rule of Validator plug-in.

Procedure to develop and use Custom validator rule in Validator plugin :-

Step-I:- Develop Java class in WEB-INF/classes folder having a java code based server-side Form Validation logic.

// CustRule.java (WEB-INF/classes)

Reffer suplimentary handout given on 12/12/12

Step-II :- Add <Struts-home>\lib\commons-validator.jar file to CLASSPATH environment Variable. as additional jar files. and compile the above source file.

Step-III :- Configure Custom Validator rule in Validator-rules.xml

in Validator-rules.xml

```
<form-validation>
  <global>
    <validator  name = "custrule"  → rule name
                classname = "CustRule"  → class name
                method = "validateCustRule"  → method name
                methodParams = "java.lang.Object,
                org.apache.commons.validator.ValidatorAction,
                org.apache.commons.validator.Field,
                org.apache.struts.action.ActionMessages,
                org.apache.commons.validator.Validator,
                javax.servlet.http.HttpServletRequest"
                } method parameters
                msg = "errors.custrule"  → key holding default error message
                jsFunction = "validateCustRule" />
                                  → JavaScript function name
```

Step-IV's Develop validateCustRule.js file having JavaScript code

validateCustRule.js (WEB-INF/classes)

refer Suplimentary handout given on 12/12/12

```
>>>>>>>>>>Struts App by using all the 3 approaches to mix up our programatic form validation logics with
        Validator plugin Declarative logics >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
-----------------------register.jsp-----------------------
<%@taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>

<html>
  <head>
   <script language="JavaScript">
       function myValidate(frm)
        {
           var flag=validateRf(frm);  //Validator plugin generated function
           frm.vflag.value="yes"; // indicates client side validations are done
           if(flag==true)
           {
           //write Client side form validation logic(programatic)
          // to check weather username and password is having same length or not
             if(frm.username.value.length!=frm.password.value.length)
                {
                alert("username and password must have same length");
                frm.username.focus();
                 flag=false;
                }//if
             }//if
             return flag;
           }//function
       </script>
   </head>
  <html:form action = "register" onsubmit="return myValidate(this)">
   <html:javascript formName="rf"/>
   Login username<html:text property="username"/><html:errors property="username"/><br>
   Login password<html:password property = "password"/><html:errors property="password"/><br>
    <html:hidden property="vflag" value="no"/>
  <html:submit value="Login"/>
 </html:form>

  <logic:notEmpty name="msg" >
      Result is: <bean:write name="msg" />
  </logic:notEmpty>
-------------------------web.xml-----------------------
<web-app>
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.xyz</url-pattern>
  </servlet-mapping>
</web-app>
---------------------struts-config.xml------------------------
<!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
        "http://struts.apache.org/dtds/struts-config_1_3.dtd">

<struts-config>
   <form-beans>
        <form-bean name="rf" type="app.RegisterForm"/>
   </form-beans>
   <action-mappings>
     <action   path="/register" type="app.RegisterAction" name="rf" input="/register.jsp">
            <forward name="result" path="/register.jsp"/>
          </action>
   </action-mappings>
```

```
    <message-resources parameter="myfile"/>

  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
     <set-property  property="pathnames"
                value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
  </plug-in>
</struts-config>


----------myfile.properties----------
# Struts Validator Default Error Messages
  errors.required={0} is required.
  errors.minlength={0} can not be less than {1} characters.
  errors.maxlength={0} can not be greater than {1} characters.
  errors.invalid={0} is invalid.
  errors.byte={0} must be a byte.
  errors.short={0} must be a short.
  errors.integer={0} must be an integer.
  errors.long={0} must be a long.
  errors.float={0} must be a float.
  errors.double={0} must be a double.
  errors.date={0} is not a date.
  errors.range={0} is not in the range {1} through {2}.
  errors.creditcard={0} is an invalid credit card number.
  errors.email={0} is an invalid e-mail address.

  #user-defined msgs supplying arg values to default error msgs
  my.un=Login username
  my.pass=Login password
  my.pass1=password
  my.msk.msg=username must contain  only alphabets
  my.unpwd.msg=username and password must have same length
  errors.custrule={0} must have same first and last characters
  # apply styles on error msgs
  errors.header=<font color=red size=2>
  errors.footer=</font>
 ----------validation.xml----------
  <!DOCTYPE form-validation PUBLIC
    "-//Apache Software Foundation//DTD Commons Validator Rules Configuration 1.3.0//EN"
    "http://jakarta.apache.org/commons/dtds/validator_1_3_0.dtd">

<form-validation>
   <formset>
      <form name="rf">
         <field     property="username"   depends="required,mask,custrule">
           <arg0 key="my.un"/>
            <msg name="mask" key="my.msk.msg"/>
            <var>
              <var-name>mask</var-name>
               <var-value>^[A-Za-z]*$</var-value>
            </var>
         </field>

         <field    property="password"  depends="required,custrule">
            <arg0 key="my.pass"/>
         </field>
      </form>
   </formset>
</form-validation>

----------validator-rules.xml----------
<!DOCTYPE form-validation PUBLIC
      "-//Apache Software Foundation//DTD Commons Validator Rules Configuration 1.3.0//EN"
      "http://jakarta.apache.org/commons/dtds/validator_1_3_0.dtd">
<form-validation>
   <global>
      <validator name="custrule"
            classname="CustRule"
            method="validateCustRule"
            methodParams="java.lang.Object,
               org.apache.commons.validator.ValidatorAction,
               org.apache.commons.validator.Field,
```

```
                org.apache.struts.action.ActionMessages,
                org.apache.commons.validator.Validator,
                javax.servlet.http.HttpServletRequest"
            msg="errors.custrule"
            jsFunction="validateCustRule"/>
  .
  .
  .
  .
 </global>
 </form-validation>
 ------------------------------CustRule.java-------------------
 //CustRule.java (WEB-INF\classes)
import java.io.Serializable;
import java.util.StringTokenizer;
import javax.servlet.http.*;
import org.apache.commons.validator.*;
import org.apache.commons.validator.util.ValidatorUtils;
import org.apache.struts.action.*;
import org.apache.struts.validator.*;

public class CustRule
{
 public static boolean validateCustRule(Object bean, ValidatorAction va, Field field,
                    ActionMessages errors, Validator validator, HttpServletRequest request)
   {
   // get FormBean property value
    String  value=ValidatorUtils.getValueAsString(bean,field.getProperty());
    // get first and last chars
     char fchar=value.charAt(0);
     char lchar=value.charAt(value.length()-1);

    // check weather first and last characters are matching or not
      if(fchar!=lchar)
       {
        errors.add(field.getKey(), Resources.getActionMessage(validator, request, va, field));
         return false;
       }
       else
       {
       return true;
       }
 }//method
}//class
//>javac CustRule.java

--------------------validateCustRule.js-----------------------------
//develop this code by taking validateRequired.js file as reference file
      function validateCustRule(form) {
            var isValid = true;
      var focusField = null;
      var i = 0;
      var fields = new Array();

      var oRequired = eval('new ' + jcv_retrieveFormName(form) +  '_custrule()');

      for (var x in oRequired) {
         if (!jcv_verifyArrayElement(x, oRequired[x])) {
            continue;
         }
         var field = form[oRequired[x][0]];

         if (!jcv_isFieldPresent(field)) {
            fields[i++] = oRequired[x][1];
            isValid=false;
         }
                  else if ((  field.type == 'text' ||
            field.type == 'textarea' ||
            field.type == 'password')) {
             //gathers comp value
                     var value = field.value;
```

```
                    //client side form validation logic
                    var fchar=value.charAt(0);
                    var lchar=value.charAt(value.length-1);

            if (fchar!=lchar) {
               if ((i == 0) ) {
                  focusField = field;
               }//iff
               fields[i++] = oRequired[x][1];
              isValid = false;
            }//if
          }//else

      }//for

           if (fields.length > 0) {
        jcv_handleErrors(fields, focusField);
      }//if
      return isValid;
   }//fx

 ----------RegisterForm.java----------
 package app;
import org.apache.struts.validator.*;
import org.apache.struts.action.*;
import javax.servlet.http.*;


public class RegisterForm extends ValidatorForm
{
  private String username="raja";
   private String password="hyd";

 public RegisterForm()
 {
     System.out.println("RegisterForm: 0-param constructor");
 }
 public void setUsername(String username)
 {
     System.out.println("RegisterForm:setUsername(-)");
     this.username = username;
 }
 public String getUsername()
 {
     System.out.println("RegisterForm:getUsername()");
     return username;
 }
 public void setPassword(String password)
 {
     System.out.println("RegisterForm:setPassword(-)");
     this.password = password;
 }
     public String getPassword()
     {
 System.out.println("RegisterForm:setPassword(-)");
          return password;
     }
     public ActionErrors validate(ActionMapping mapping,HttpServletRequest req)
     {
          // read vflag value
     String vstatus=req.getParameter("vflag");

     ActionErrors errs=new ActionErrors();
     if(vstatus.equals("no"))  // when client side form validations are not done
     {
      //performs server side validator plugin based form validations
       errs=super.validate(mapping,req);
     //write Server side form validation logic(programatic)
     // to check weather username and password is having same length or not
     if(errs.size()==0)
     {
      if(username.length()!=password.length())
```

```
289           {
290            errs.add("password",new ActionMessage("my.unpwd.msg"));
291           }//if
292          }//if
293
294         }//if
295         return errs;
296      }//validate(-,-)
297   }//class
298   //>javac -d . RegisterForm.java
299   ----------RegisterAction.java----------
300   package app;
301   import org.apache.struts.action.*;
302   import javax.servlet.http.*;
303   import javax.servlet.*;
304
305
306   public class RegisterAction extends Action
307   {
308     public RegisterAction()
309     {
310        System.out.println("RegisterAction: 0-param constructor");
311     }
312     public ActionForward execute(ActionMapping mapping,ActionForm form,HttpServletRequest req,HttpServletResponse res)t
313     {
314        System.out.println("RegisterAction:execute(-,-,-,-)");
315        RegisterForm rf = (RegisterForm)form;
316        String user = rf.getUsername();
317        String pass = rf.getPassword();
318        if(user.equals("Sathya") && pass.equals("tech"))
319        {
320        req.setAttribute("msg","Valid Credentials");
321        }
322        else
323        {
324        req.setAttribute("msg","InValid Credentials");
325        }
326        return mapping.findForward("result");
327     }
328   }
329
```

Step-II :- add default an error message in properties file

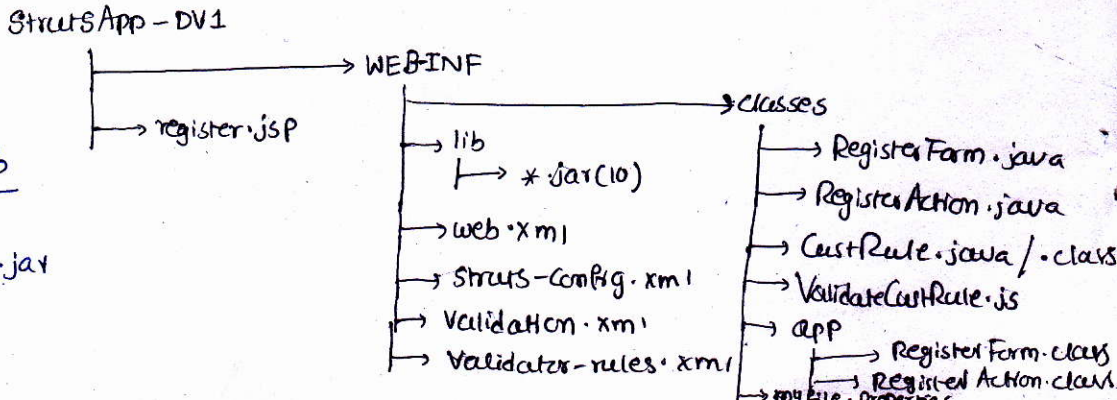
in myfile.properties

errors.custrule = {0} must have same first and last characters

Step-III :- Configure this rule on FormBean properties through Validation.xml file.

```
<form-validation>
  <formset>
    <form name="rf">
      <field property="username" depends="requiered, mask, custrule">
        <arg0 key="my.un" />
        <msg name="mask" key="my.msk.msg" />
        <var>
          <var-name> mask </var-name>
          <var-value> ^[A-Za-z]*$ </var-value>
        </var>
      </field>

      <field property="password" depends="required, custrule">
        <arg0 key="my.pass" />
      </field>
    </form>
  </formset>
</form-validation>
```

Deployment Directory structure of the above steps based appn.



Jar files added to classpath

Struts-core-1.3.8.jar

Commons-validator-1.3.1.jar

Servlet-api.jar

Conclusion:- To add our programatic FormValidation logics to execute along with Validator Plugin logics. there are 3 approaches as discussed above.

→ approach① is specific to one FormBean property on which it is applies

→ approach② is visible in multiple FormBean property of single FormPage / FormBean but it can use multiple FormBean properties data while developing form validation logic. (Can't work with multiple FormBean properties)

→ approach③ is visible in multiple FormBeans / FormPages but can coork with only one FormBean property data at a time.

13/12/2012

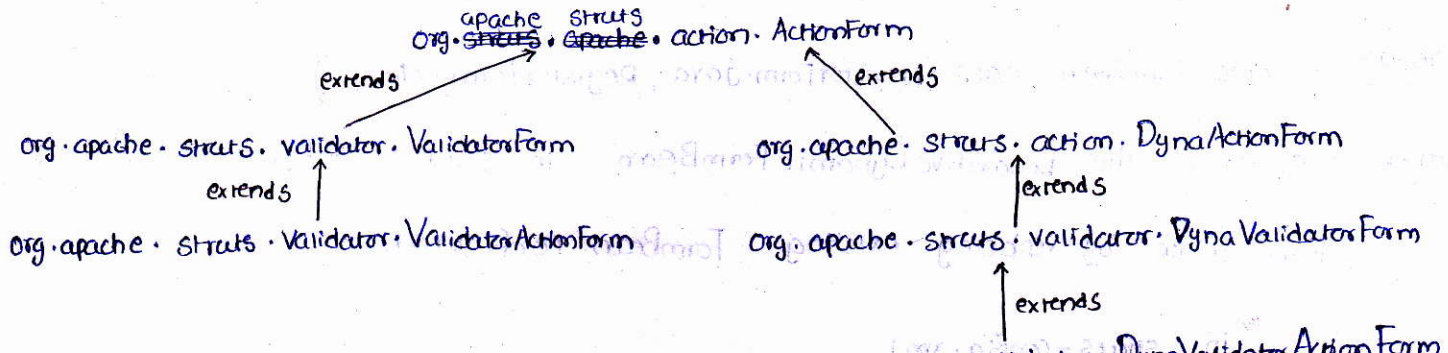
→ From struts 1.3 we can place <arg> tag instead of arg0, arg1, arg2, ... tags, when <arg> tag is used the values to arguments will be supplied in the order the <arg> tags are specified under <field> tag of validation.xml

Eg:-

```
<field property="username" depends="required, minlength">
    <arg key="my.un"/>  → supplies {0} value of required, minlength rule msgs.
    <arg name="minlength" key="${var:minlength}" resource="false">
        <var>
            <var-name> minlength </var-name>
            <var-value> 5 </var-value>
        </var>
</field>
```

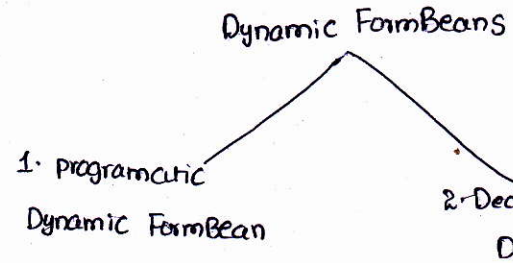
Supplies {1} value of minlength rule msg

→ Important FormBean Types



→ The FormBean whose FormBean properties, getter, setter methods are generated dynamically by ActionServlet is called dynamic FormBean.

→ To work with Dynamic FormBeans we must deal with DynaXxxForm classes of Struts api.



→ programmers explicitly develops FormBean class with form bean properties setXxx(-) and getXxx() methods

→ Possibility is there to place reset(-) validate(-,-) methods in FormBean class

→ It is suitable to perform both programatic and Declarative Form Validations

→ We never take separate FormBean class here. but we configure DynaXxx Form classes directly in Struts Config file

→ No Possibility is there to place reset(-,-), validate(-,-)

→ It is suitable to perform only Declarative (validator plugin based) FormValidations

NOTE:- Programatic Dynamic FormBeans are recomended to use

→ DynaActionForm class can't be used for validator plugin based Declarative Form Validations.

→ But DynaValidatorForm, DynaValidatorActionForm classes can be used for DeclarativeFormValidation

Procedure to place declarative DynamicFormBean in our 1st Struts application. (StrutsApp)

Step-1:- Delete FormBean class (RegisterForm.java, RegisterForm.class)

Step-2:- Configure the Declarative DynamicFormBean in Struts configuration file as shown below by replacing existing FormBean configuration

in struts-config.xml

```
<struts-config>
  <form-beans>
    <form-bean name="rf" type="org.apache.struts.action.DynaActionForm
      <form-property name="username" type="java.lang.String" initial="kiran" />
      <form-propery name="passcoord" type="java.lang.String" />
    </form-bean>
  </form-beans>

  <action-mappings>
    <action>
      =
      =
    </action>
  </action-mappings>
</struts-config>
```

must match with FormComponent names

Step-3:- Write following code in Actionclass to read the FormData from the FormBean class object.

in execute(-,-,-,-) method of RegisterAction class

```
public ActionForward execute(-,-,-,-)
{
//read form data from FormBean class obj
DynaActionForm fm = (DynaActionForm) form;
String user = (String) fm.get("username");
String pass = (String) fm.get("passcoord");
                                  FormBean properties
  =
  =
}
```

Step-4 :- execute the application in Reguler manner

NOTE :- Since we are not developing seperate FormBean manual FormBean class while working with Declarative DynamicFormBean. so there is no provision for programa to place reset(-,-), validate(-,-) methods.

*) Procedure to work with programatic Dynamic FormBean to work with all kinds of FormValidations in StrutsApp-DV appn.

Step-1 :- keep StrutsApp-DV appn ready. (refer handout given on 12/12/12).

Step-2 :- make the FormBean class extending from DynaValidatorForm class → remove FormBean properties, setXxx(-) methods as shown below.

```
// RegisterForm.java
    = } package imports

public class RegisterForm extends DynaValidatorForm
{  // do observe that no form bean properties, getXxx(), setXxx(-) methods are replaced here.
   public ActionErrors validate(ActionMapping mapping, HttpServletRequest )
   {
     //read form data
   String username = (String) get("username");
   String password = (String) get("password");

        = } same as 276 to 295 of validate(-,-) [refer 12/12/12 handout]

   }//validate(-,-)
}//class
```

Step-3 :- add <Struts_home>\lib\commons-Beanutils-1.7.0.0 jar file to classpath env Variable and Compile the above FormBean class.

Step-4 :- Configure the above FormBean class in Struts configuration file as shown below. in Struts-config.xml

```
<struts-config>
   <form-beans>
```

```
<form-bean   name="rf"   type="app.RegisterForm">
    <form-property   name="username"   type="java.lang.String"   initial="kiran"/>
    <form-property   name="password"   type="java.lang.String"/>
</form-bean>
</form-beans>
        =
        =
        =
        =
</struts-config>
```

Step-5:- ~~Run the application in regular manner.~~ Write following code in Action class execute(-,-,-,-) to read the FormData.

```
public  RegisterAction()
  {
     s.o.p ( );
  }
public  AF  execute(-,-,-,-).
  {
     RegisterForm   rf = (RegisterForm) form;
     String   user = (String) rf.get("username");
     String   pass = (String) rf.get("password");
              =
              =
              =
  }
}
```

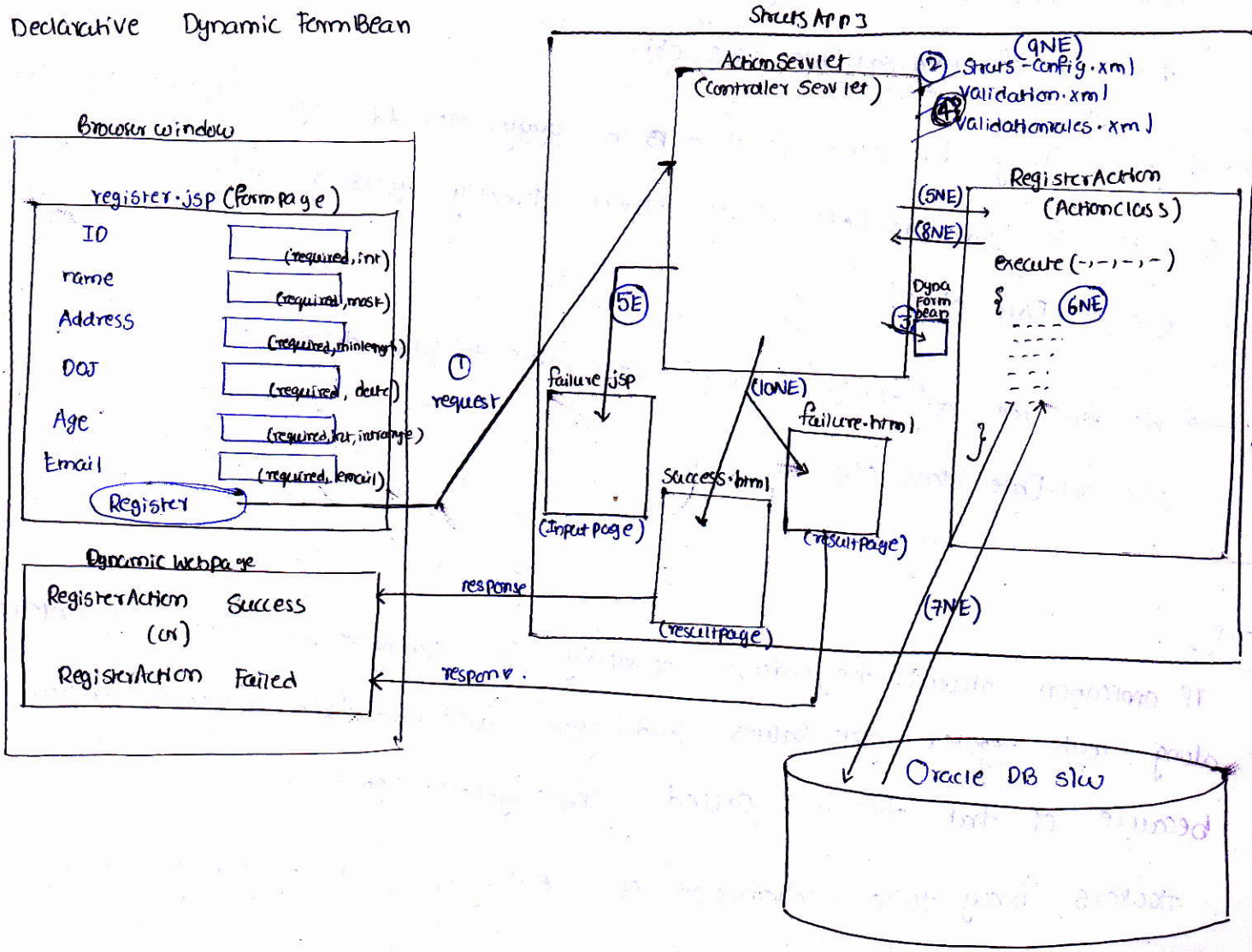
NOTE:- Since we develop programatic DynamicFormBean through manually develop FormBean class so we get facility to place Validate(-,-), reset(-,-) methods.

Step-6:- Run the appn in regular manner.

14/12/12

→ If we are just looking to apply Declarative form validations (Validator plugin) It is recommended to use Declarative Dynamic form beans. coe are planning to apply both programatic and Declarative validations then it is recomended to use programatic Dynamic form bean.

(*) Example application to work coith validation rules of Validator plug-in and Declarative Dynamic FormBean



In the above diagram 4? mark indicates validators plug-in based FormValidations

If FormValidation errors are generated the control goes to the input page failure.jsp Other wise control goes to execute(-,-,-,-) of Action class.

For the above diagram based application refer appn (4) of (26) to (30) pages

Validate = false kept in <action> makes ActionServlet not to call the validate() of FormBean class in FormBean lifecycle.

That means using this we can disable server side validations in our Struts application. The default value of this ~~value of~~ validate attribute is true indicating Server side validations are always enable.

15-12-2012

→ To insert value to date data type db table column we must take date value in the form of java.sql.Date class obj.

→ if given String date value is there in yyyy-MM-dd pattern then it can be converted in to java.sql.Date class object directly by using valueOf(-) of java.sql.Date class.

→ We can use setDate(-,-) to give date value to jdbc driver in the form of java.sql.Date class obj

---

Imp ****

If application allows the enduser to supply SQL query related technical content along with regular input values and if your application data, flow is disturbed because of that then it is called "SQL Injection problem"

Hackers may take advantage of SQLInjection like getting into application by submitting wrong password or no password

While working with Login application where Simple Statement object is used we can notice this SQLInjection problem.

Username : raja' --
→ correct username

Password : kkkk
→ wrong password

o/p ⇒ Valied credentials
→ wrong output

Since Simple Statement object sends query to database slw and compiles query in DB slw along with given input values. so, if any SQL Technical Content (↳ SQL comment (--)) appended to input values the DB compiler recognizes them and this may change the original behaviour of the query.

To solve SQL Injection problem use prepared Stmt Object instead of Simple Statement object. prepared Statement object sends query to database slw & Compiles query in DB slw without input values. But input values will be set to query after Compilation, just before execution. so, the SQL technical Content appended with application input values will not be recognized as SQL Content. so, there is no possibility of getting SQL injection problem

Q:- Can we develop the Struts application without FormBean?

Ans:- Yes, we can develop but not recomended.

If Struts Action class is getting request from hyperlink then FormBean is not required

If Struts Action class is getting request from FormPage then also we can Avoid FormBean class but to perform FormValidations effectively it is recommended to take FormBean class

17/12/2012

MyEclipse = Eclipse + Built-in plugins

These plugins simplify the Advanced technologies based project development.

MyEclipse

type: IDE for java Environment

Version: 8.x (Compatible with jdk 1.5+)
10.x (Compatable with jdk 1.6+)

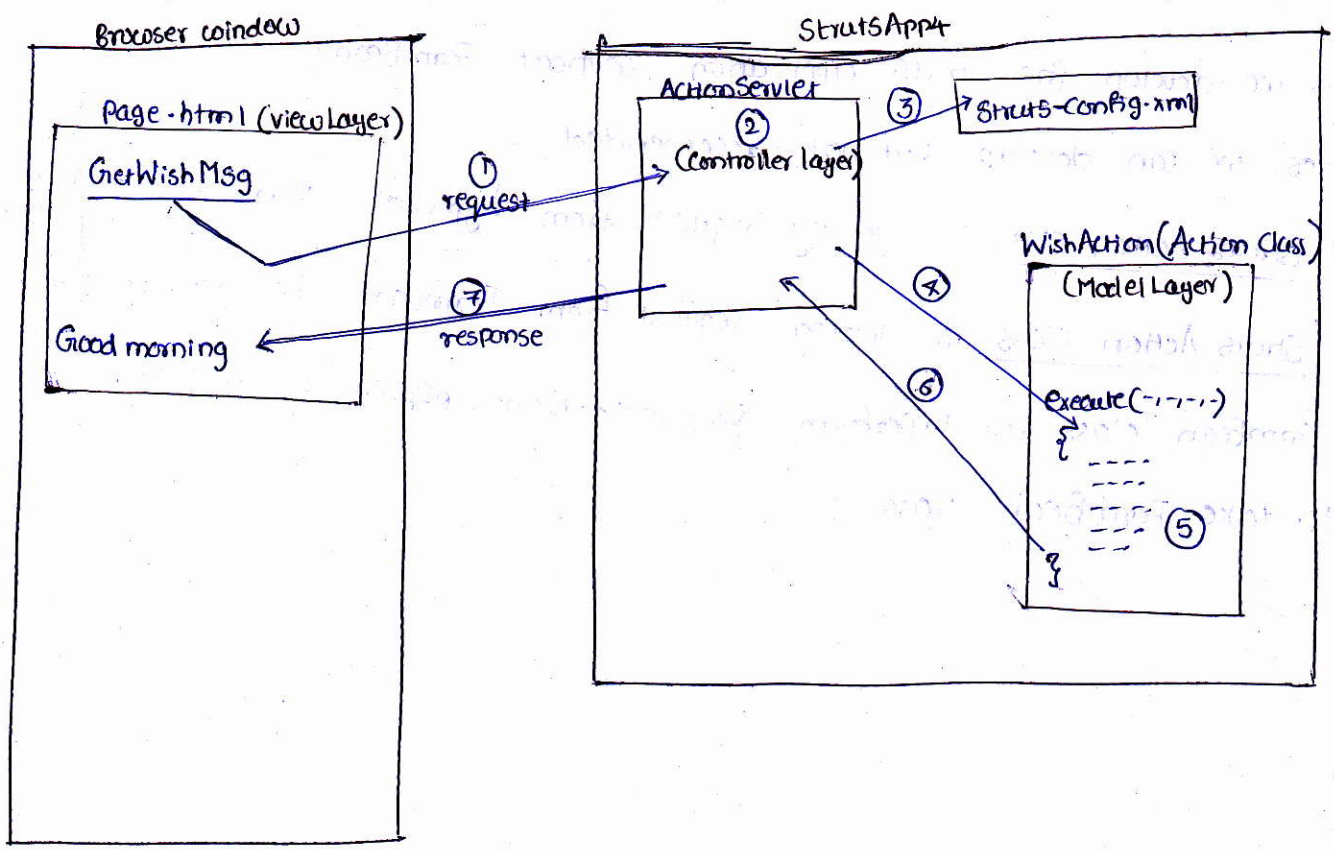
Vender: Eclipse org

Commercial IDE

given Tomcat as built-in server and also to cfg other external server

To download slw: www.myEclipseIDE.com

for Docs: www.myeclipseide.com



procedure to develop the above application by using MyEclipse IDE.

Step-I:- launch MyEclipse IDE by choosing Workspace folder.

↳ It is the folder where projects will be save

Step-II:- Submit the Subscription details

MyEclipse Menu ⟶ Subscription information ⟶ Subscriber: Sathya Technologies
Subscription code: FLR8ZC-8S5-55-696650

5938947 27

Step-III :- Create Web project in MyEclipse IDE.

File Menu → New → Web project → projectName: Struts App4 → Finish

Step-IV :- add Struts capabilities to the project.

Right click on project → MyEclipse → Add Struts Capabilities → Struts 1.3 → Base package for new classes: P1 → finish.

The above steps add Struts jar files to the libraries of the project.

Step-V :- add Action class to the project

Right click on Src folder → New → Other → MyEclipse → Web-Struts → Struts 1.3 → Struts 1.3 Action → Next → path: /wpath →

Type: WishAction → forwards tab → add → Name: success → add → close

path : /page.jsp

→ finish

Step-VI :- Write following code in the execute(-,-,-,-) of Action class. (WishAction)

ctrl+shift+O for package import stmts

```
public class WishAction extends Action {
    public ActionForward execute(-,-,-,-) {
        //Write b.logic
    Calender cl = Calender.getInstance(); // gives sys date and time
    int h = cl.get(Calender.HOUR_OF_DAY); // gives current hour
        // generate wish msg
    String msg=null;
    if (h<=12)
        msg = "Good morning";
    else if (h<=16)
        msg = "Good After noon";
    else if (h<=20)
        msg = "Good Evening";
    else
        msg = "Good Night";
    request.setAttribute("WishMsg", msg);
```

return mapping.findForward("success");

} //execute(-,-,-,-)

} //class

Step-VII: add page.jsp to the project.

Right click on Webroot folder → new → JSP → File Name: page.jsp → Finish

<%@taglib uri="http://struts.apache.org/tags-html" prefix="html" %>

<%@taglib /tags-bean" prefix="bean" %>

<%@taglib /tags-logic" prefix="logic" %>

<html:link action="wparb"> Generate Wish Msg </html:link>

<logic:notEmpty name="wishmsg">

<bean:write name="wishmsg" />

</logic:notEmpty>

Step-VIII :- Configure External Tomcat Server with MyEclipseIDE

Window Menu → preference → MyEclipse → servers → Tomcat →

Tomcat6.0 → ⊙Enable → Tomcat-home directory (browse) → Apply → ok

Step-IX:- Run the project

Right click on project → Run as → MyEclipse Server app → Tomcat 6.0 → ok

Step-X:- Test the appn click the above Globab symbol → Type url

http://localhost:2020/StrutsApp4/~~wparb.do~~ page.jsp

Generate Wish msg

NOTE:- <html:link> tag is alternate for the traditional <a> tag

"href" attribute can't generate extentions dynamically in the given uri but action attribute can generate.

(*) procedure to configure the Domain Server of Weblogic 10.x in MyEclipse IDE

Step-I:- Create Domain Server in Weblogic 10.x

Start → Programs → Oracle Weblogic → Quick Start → ☑ Getting Started with Weblogic Server

→ Next → Next → Domain Name: [Strurs Domain] → Next →

username: [javaboss]

User password: [javaboss1]

Confirm Password: [javaboss1]

→ [✓] Adiminstration Server → Next → Listenport [7005]

→ Next → create → done

Step-II:-

~~procedure to~~ Configure weblogic 10.x Server with MyEclipse IDE.

Window Menu → preferences → MyEclipse → Servers → Weblogic →

Weblogic 10.x → (•) Enable → BEA home directory [c:\Oracle\Middleware] [Brows.]

↳ installation folder

→ Administration Username: [Javaboss] → Administration pwd: [javaboss1]

Execution domain root: [c:\Oracle\middleware\user_projects\domain\StrursDomain] [Brows]

↳ The folder representing our domain

→ Apply → Ok

~~Configuring JBoss server 5x with MyEclipse IDE~~

(*) procedure to Configure JBoss 5.x default domain Server with MyEclipse IDE

port in Server.xml

Window Menu → preferences → MyEclipse → Servers → JBOSS →

JBOSS 5.x → (•) Enable → JBOSS home directory: [E:\Jboss 5.x soft\jboss-5.1.0GA] [Brows]

→ apply → Ok

(*) Procedure to Configure Glassfish 2.x Server with MyEclipse IDE

see port no in Domain.xml

Window Menu → preferences → MyEclipse → Servers → Glassfish →

Glassfish 2.x → (•) enable → Home directory: [c:\sun\Appserver] [Browse]

→ apply → Ok

NOTE:- Every domain Server is an application Server. if Company uses Same appn server s/w in multiple projects then multiple domain Servers will be created in that ~~the~~ appn Server s/w. we can't create domains in tomcat. default domains are not there in Weblogic 10.x. But we can create user define domains.

Default domains in JBOSS are :-

Standard
default (default)
web
minimal
all

Default domain in GlassFish is : domain1

18/12/2012

There are built in Actions given by Struts API. we can use these Action classes directly in Struts Configuration file or we can use these Action classes in development of our Struts Action classes.

The Built-in Action classes are :-

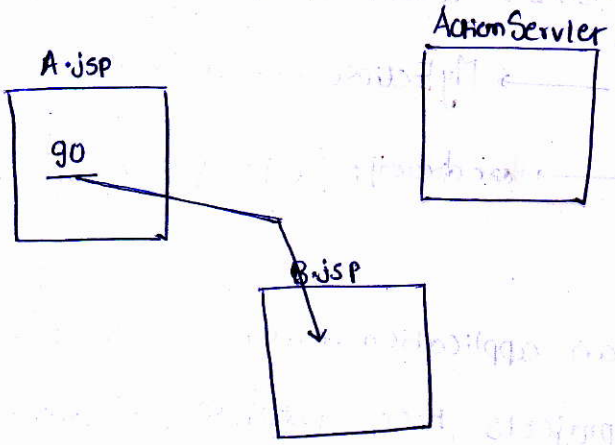
- SwitchAction
- DispatchAction
- LookupDispatchAction
- MappingDispatchAction
- LocaleAction
- DownloadAction
- and etc...

} available in org.apache.struts.actions.Action

→ All these classes are the sub classes of org.apache.struts.actions.Action (C)

→ All these classes are available in struts-extras-1.3.8.jar file
(<struts_home>\lib folder)

→ Problem :-



go

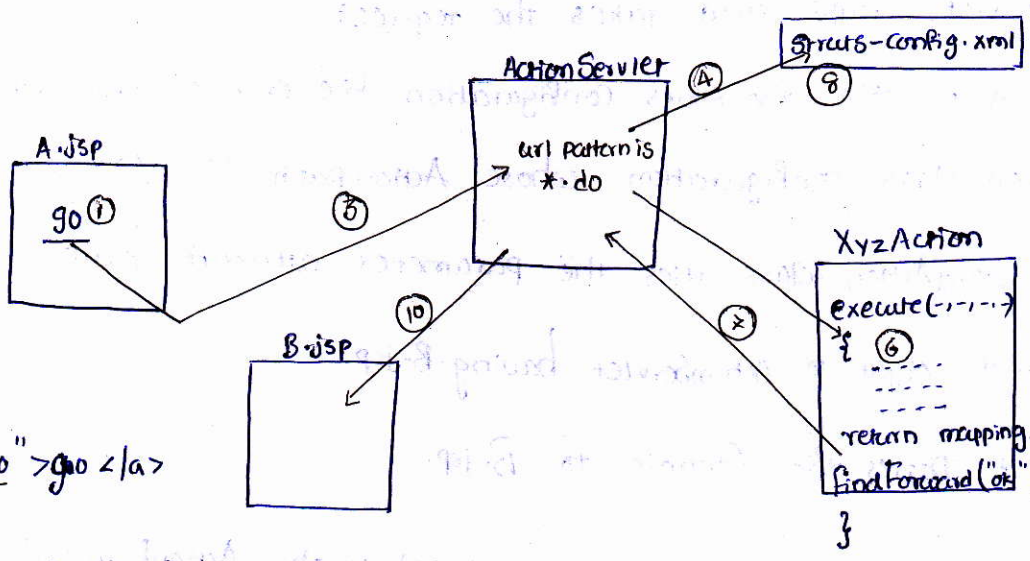
(or)

<html:link

According to MVC2 rules two Jsp's of web application must not interact with the directly. that means they must interact through Controller Servlet (i.e Action Servlet)

In the above diagram MVC2 rule is violated because A.jsp is directly interact with B.jsp

Solution 1 :-



A.jsp

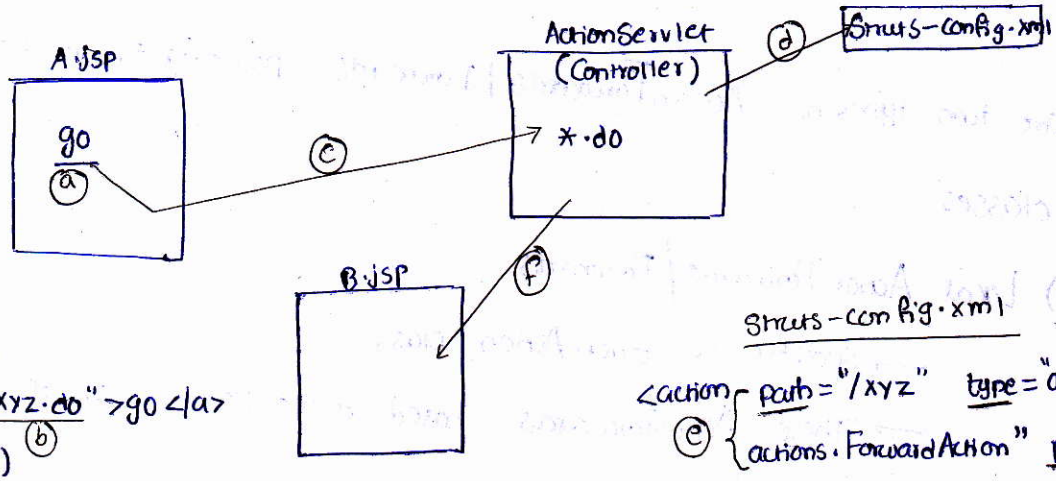
```
<a href="xyz.do">go</a>
(or)
<html:link href="xyz.do">
go
</html:link>
```

struts-config.xml

```
<action path="/xyz" type="XyzAction">
<forward name="ok" path="/B.jsp" />
</action>
```

here A.jsp is passing the control to B.jsp through ActionServlet and separate Action class XyzAction and also satisfying MVC2 rules. But taking one separate userdefined Action class to pass controle from one jsp to another jsp is not recomended.

Solution 2 :- (using org.apache.struts.actions.ForwardAction class)



A.jsp

```
<a href="xyz.do">go</a>
(or)
<html:link href="xyz.do">go</html:link>
```

struts-config.xml

```
<action path="/xyz" type="org.apache.struts.actions.ForwardAction" parameter="/B.jsp"/>
```

NOTE:- make sure that struts-extras-1.3.8.jar file is added to the WEB-INF/lib Folder.

In the above diagram MVC2 rule is not violated, because the A.jsp is talking with B.jsp through the Controller Servlet called ActionServlet by using

ForwardAction class. w.r.to the diagram (a), (b) the hyperlinks generates the request to Struts application based on "href" url.

(c) ActionServlet traps and takes the request

(d) ActionServlet uses the Struts configuration file entries and looks for the ForwardAction class configuration whose Action path is "/xyz".

(e) This ForwardAction class uses the parameter attribute value and returns ActionForward object to ActionServlet having B.jsp.

(f) ActionServlet passes the Controle to B.jsp

NOTE:- ForwardAction class is no way related with ActionForward class. ForwardAction is required to navigate the Controle blw the resources of application through Controller Servlet.

Where as ActionForward is required to locate the result pages of Action class.

→ Solution 2 is recomended to use, because there is no need of taking Separate Userdefined Action class.

There are two types of ActionForwards | Forwards pointing to result page of Action classes

1) Local ActionForwards | Forwards
   → specific to Each Action class
   → These ActionForwards based result pages can be used only by one Action class.
   → place <forward> under <action> for this.

2) Global ActionForwards | Forwards
   → Common for all the Struts Action classes of Struts application

→ These ActionForwards based Result pages can be used by multiple Action classes.

→ use <global-forwards>, <forward> tags for this.

Struts Configuration file

```
<struts-config>
    <form-beans>
        .....
        .....
        .....
    </form-beans>
    <global-forwards>
        <forward name="r1" path="/result1.jsp" />
        <forward name="r2" path="/result2.jsp" />
    </global-forwards>
    <action-mappings>
        <action path="/xyz" type="XyzAction1">
            <forward name="r3" path="/result3.jsp" />
        </action>
        <action path="/abc" type="ABCAction1">
            <forward name="r4" path="/result4.jsp" />
        </action>
    </action-mappings>
```

} global forward Configuration.

↳ LocalForward Configuration

↳ Local Forward Configuration

XyzAction1 class can use result1.jsp, result2.jsp and result3.jsp prgs as result pages.

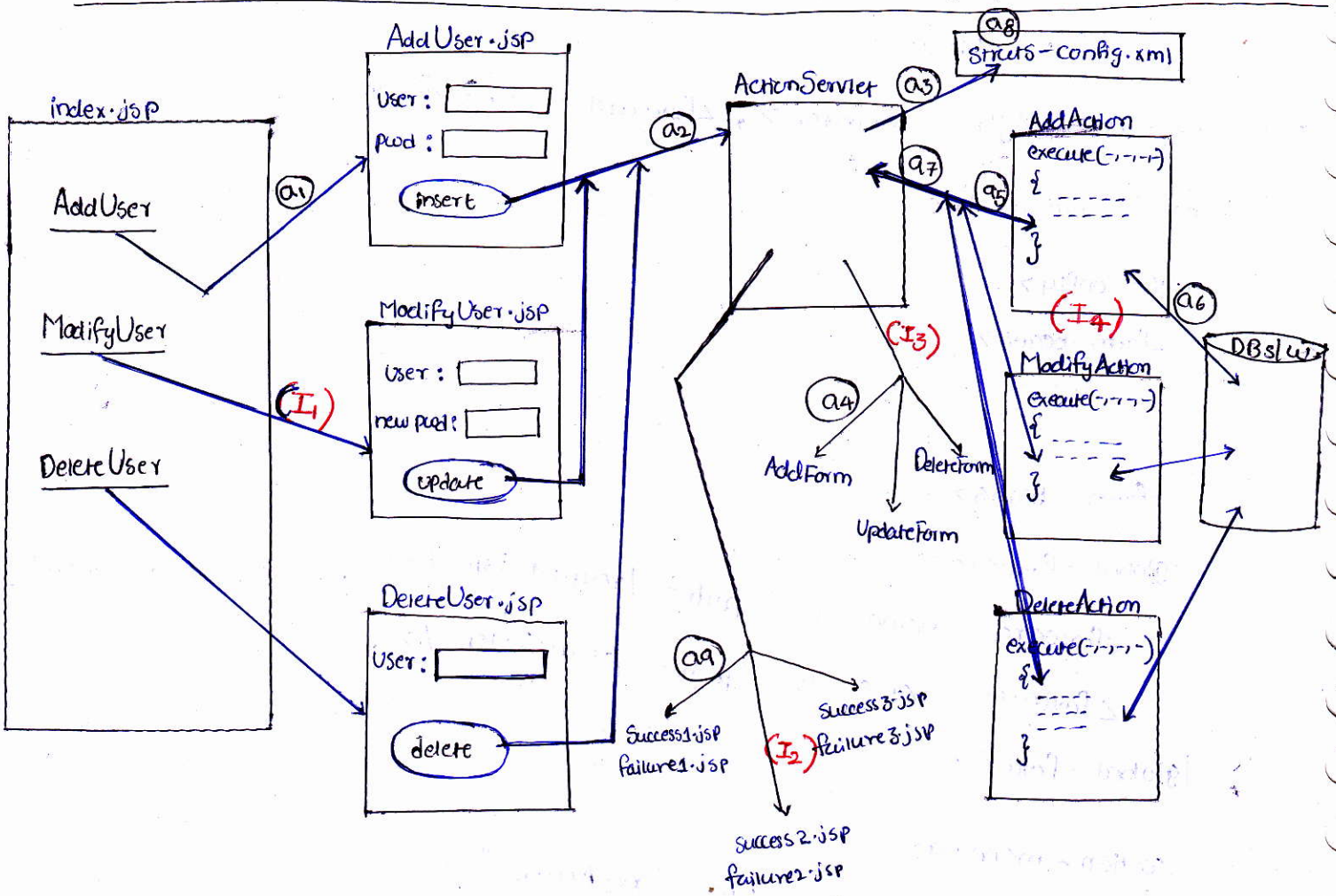
ABCAction1 class can use result1.jsp, result2.jsp and result4.jsp prgs as result pages.

Q:- What happens if Local forward Configuration and global Forward Configuration is done by having same Logical name & pointing to two result pages?

Ans:- When Action class returns ActionForward Object having the above specified logical name then the result page configure in LocalForward Configuration will be taken as the result page of StrutsAction class.

19/12/2012

Developing the struts application having multiple Form pages, FormBeans, Action classes



The above diagram based web application can be improvised in 4 areas for betterment

(I1) index.jsp is interacting with other jsp's directly. which is against MVC2 rule. so, use ForwardAction class for this navigation

(I2) instead of taking separate LocalActionForwards based result pages for every Action class it is recomended to use common result pages for all ActionClasses through GlobalForwards

(I3) Since all the three form pages are dealing with Similer formdata we can take single formBean class instead of multiple FormBean classes

(I4) Three Actionclasses are having much Similler logics so, we can Combined them into Single Action class by taking the support of DisparchAction class

The Java class that extends from org.apache.struts.actions.DispatchAction class is called DispatchAction class. (Abstract class with no Abstract methods)

→ This class can have multiple request processing logics in multiple user defined method definitions.

→ These methods signature, return type must match with execute(-,-,-,-) method

→ Every request i.e. coming to DispatchAction class must carry method name as additional request parameter value. In order to execute that method of DispatchAction class for processing request. That the name of this Request parameters will be choosen by programmer as "parameter" attribute value of <action> tag while configuring DispatchAction class in Struts Configuration file.

→ When multiple Form pages are planning to use multiple methods of DispatchAction class to process the request then these Formpages must send the method names of DispatchAction class along with generated request as additional request parameter value.

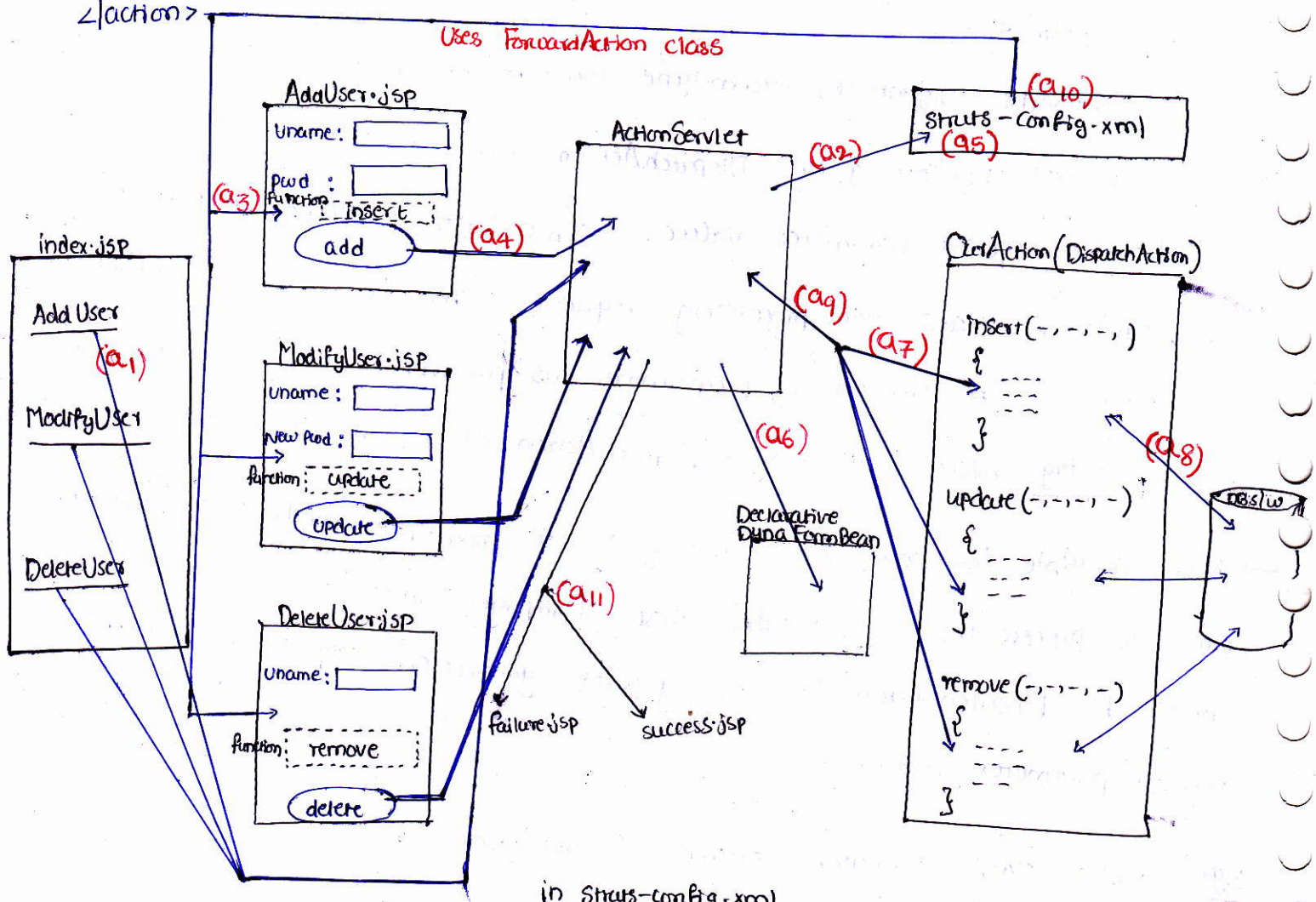
Eg:-
```
public class OurAction extends DispatchAction
{
  public ActionForward insert(-,-,-,-)
  {
    ------
    ------ // request processing logic
    ------
  }
  public ActionForward update(-,-,-,-)
  {
    -----
    ---- // request processing logic
    ----
  }
  public ActionForward remove(-,-,-,-)
  {
    -----
    ---- // request processing logic
    ----
  }
}
```

in Struts-Config.xml

<action path="/xyz" type="OurAction" name="rf" parameter="function">

(additional request parameter name.)

: New Impravise Diagram :



in Struts-config.xml

<action path="/xyz" type="OurAction" parameter="function">

</action>

We generally used hidden boxes in our Form pages to send the method names of DispatchAction class as additional request (function) parameter names along with the Form pages generated request. as shown above. For this the names of this hiddenboxes must be collected from DispatchAction class configuration related additional request (function) parameter name.

W.r.to the diagram

(a1) end user clicks on AddUser hyper link

(a2) ActionServlet traps and takes the request

(a3) ActionServlet uses ForwardAction class configuration done in Struts Configuration file to transfer the control to the form page AddUser.jsp.

(a4) EndUser submits the request to ActionServlet from AddUser.jsp. This request contains function = insert as additional request Param Value along with reguler Form data.

(a5) ActionServlet traps and takes this request and uses the Struts configuration file entries to decide the FormBean and ActionClass that are required to process the request.

(a6) ActionServlet writes received FormData to FormBean class object

(a7) ActionServlet calls execute(-,-,-,-) on our DispatchAction class object - since not available to super class (DispathAction) execute(-,-,-,-) will execute.

(a8) This SuperClass execute(-,-,-,-) reads additional request parameter Value (nothing but "insert" from "function" requestParam) and calls method of our DispatchAction class whose name is additional RequestParam Value (calls insert(-,-,-,-) method)

(a9) This insert(-,-,-,-) completes the persistence operation on DataBase

(a9) insert(-,-,-,-) returns ActionForward object to ActionServlet.

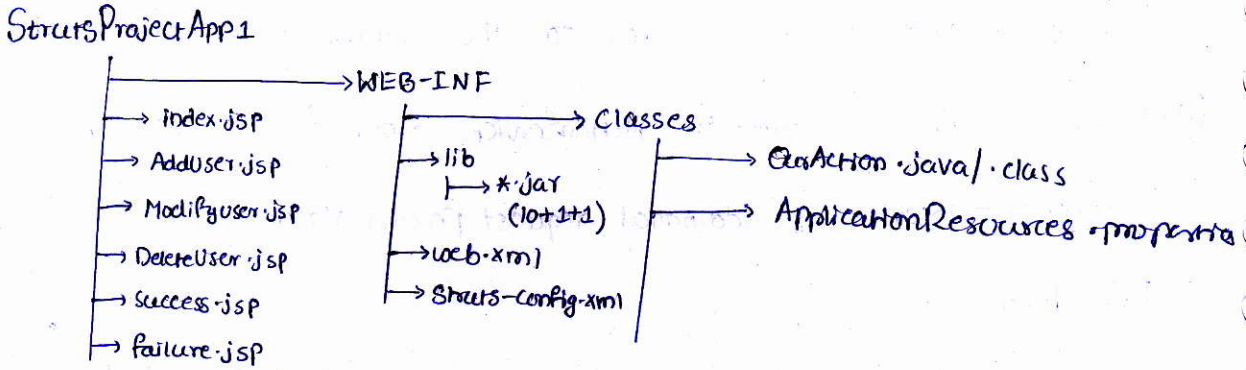
(a10) ActionServlet uses entries of Struts Configuration file to decided the result page.

(a11) ActionServlet passes the control to ResultPage.

Don't override execute(-,-,-,-) in our DispatchAction class Because it doesn't give chance to execute the Superclass execute(-,-,-,-) method.

For example application of Part-I of above miniproject refer the application (5) of page no's (33) to (37) of the booklet.

Directory Structure :-



Jar files in CLASSPATH :-

Servlet-api.jar

Struts-core-1.3.8.jar

Struts-extras-1.3.8.jar

Jar files in WEB-INF\lib folder :-

10 ⟶ Struts jar files

1 ⟶ Ojdbc14.jar

1 ⟶ Struts-extras-1.3.8.jar

Flow of execution Related to application ⓔ of execution :-

A) Programer deployees StrutsProjectApp1 application in webserver.

B) Because of <load-on-Startup> the ServletContainer performs pre-Instantiation & pre-initialization of ActionServlet either during Server startup and during the deployment of Webapplication (refer : 30.)

C) ActionServlet reads and verifies the entries of Struts configuration file because of " Step ⓑ "

D) End user gives request to Struts application

http://localhost:2020/StrutsProjectApp1

E) The welcome file Index.jsp executes (refer 38 to 40, 2-15)

F) EndUser clicks on add hyperlink of Index.jsp. (refer: 10)

G) ActionServlet traps and takes this request (refer: 23 to 36)

H) ActionServlet uses ForwardAction class configuration whose action path is

"/add" and forwards the control to "AddUser.jsp" form page.
↳ refer: 65 refer: 69 ↵

(I) The AddUser.jsp will be launch on the brocoser coindoco. This internally creates FormBean class Object (ref: 76 to 100). & (49 to 53)

(J) EndUser submits the Formpage (AddUser.jsp). This Form page sends the request to Strurs Appn. (ref: 95 & 81). This request contain function = insert as additional request param value along with reguler form data.

(K) ActionServlet traps and takes the requiest (ref: 23 to 36)

(L) ActionServlet creates FormBean class object and writes the received Formdata to it. (ref: 49 to 53)

(M) ActionServlet creates our DispatchActionClass object (ref: 62 & 63)

(N) ActionServlet calls on our DispatchAction class nothing but our ActionClass object. Since execute(-,-,-,-) is not available the Superclass execute(-,-,-,-) method will execute.

(O) This Super class execute(-,-,-,-) reads additional request parameter value (insert) and calls insert(-,-,-,-) of our ActionClass. (ref: 198 to 235)

(P) This insert(-,-,-,-) returns ActionForcoard object with logical name (success/ failure.)

Success		Failure
220 ⟶	(Q)	⟵ 224 or 229
57 ⟶	(R)	⟵ 58
311-327 (success.jsp) ⟶	(S)	⟵ 328-335 (failure.jsp)

324 ⟶ (T)

23-36 ⟶ (U) [ActionServlet traps and takes the request]

68 ⟶ (V)

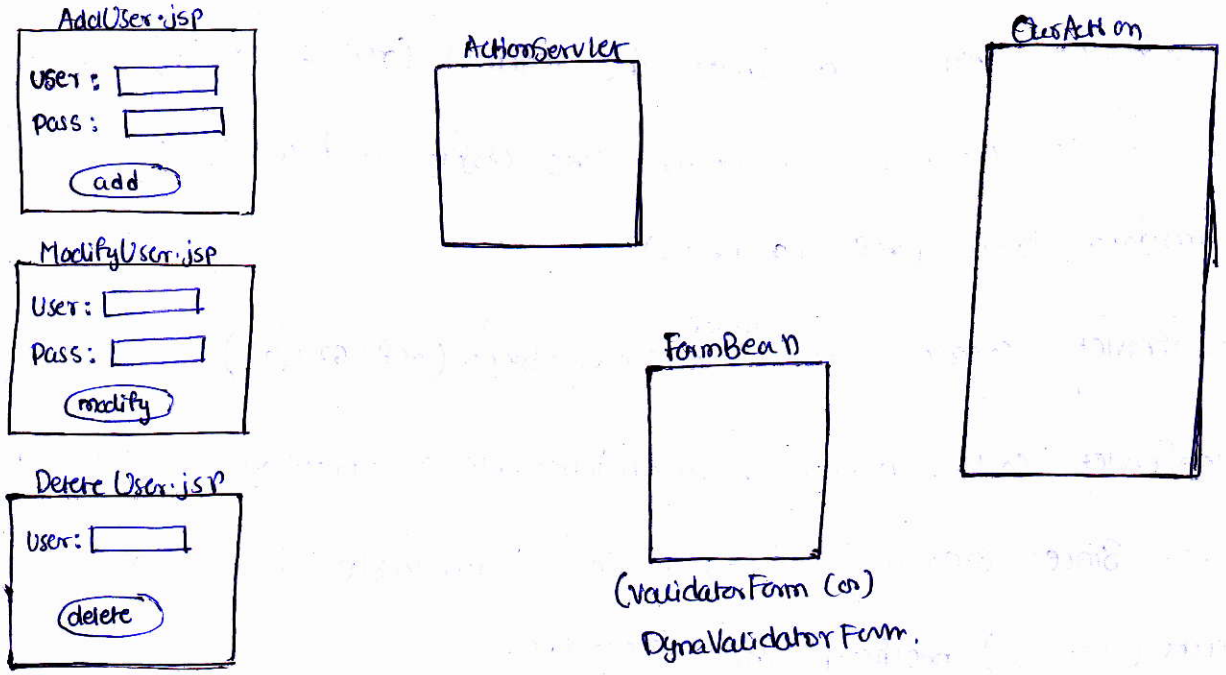
(index.jsp) 3-15 ⟶ (W)

problem towards Form Validation :-

21/12/2012

problem:- When FormBean type is ValidatorForm (or) DynaValidatorForm we need to configure validator rules in Validation.xml file by specifying FormBean logical name.

This gives one practicle problem i.e. when multiple formpages are using SingleForm Bean of type ValidatorForm (or) DynaValidatorForm then we can't write Separate and different Validator rules for each formpage.



Note:- All the 3 formpages are using single FormBean class.

→ TO Solve this problem there are two solutions using

ValidatorActionForm class
(or)
DynaValidatorActionForm class.

Solution 1: While working with ValidatorActionForm, DynaValidatorActionForm type FormBean classes we can configure validator rules on Formpages based on the target Action classes of Formpages. (not based on the FormBeans of Formpages).

for this we need to specify Action path of Action class in validation.xml while configuring validator rules for each Form page.

Solution 1 implementation :-

Step-I :- make multiple form pages utilizing Single FormBean and multiple Action classes on one Action class for Formpage basis

Step-II :- Take FormBean type as ValidatorActionForm or DynaValidatorActionForm and also make all the # Action classes to use Single FormBean.

Step-III :- in validation.xml file write separate validator rules for every Formpage by specifying the actionpath of Action class used by that Formpage.

Step-I Sample code Struts-config.xml

```
<Struts-Config>
  <Form-beans>
    <Form-bean  name="rf1"  type="org.apache.struts.validator.DynaValidatorActionForm
                  └→ FormBean logical name
      ----------
      ----------
      ----------
    </Form-bean>
  </Form-beans>

  <Action-mappings>
    <Action  path="/xyz1"  type="XyzAction1"  name="rf1">
                └→ Action path of Actionclass        └→ FormBean logical name
      -------
      -------
    </Action>
    <Action  path="/xyz2"  type="XyzAction2"  name="rf1">
                └→ Action Path of Action class       └→ FormBean logical name
      -----
      -----
    </Action>
    <Action  path="/xyz3"  type="XyzAction3"  name="rf1">
                                                 └→ FormBean logical name
      -----
      -----
    </Action>
  </action-mappings>
</Struts-Config>
```

Step-II Sample code

Form pages

AddUser.jsp

```
<html:form action="xyz1">
    ------
    ------
    ------
</html:form>
```

ModifyUser.jsp

```
<html:form action="xyz2">
    ------
    ------
</html:form>
```

DeleteUser.jsp

```
<html:form action="xyz3">
    ------
    ------
    ------
</html:form>
```

Step-III Sample Code

Validation.xml

```
<form-validation>
  <formset>
```

Validator rules Configuration for AddUser.jsp

```
    <form name="/xyz1">        (action path used by AddUser.jsp)
      <field name="id" depends="required">
        ------
        ------
      </field>
      <field name="pass" depends="required">
        ------
        ------
      </field>
    </form>
```

Validator rules Configuration for modifyUser.jsp

```
    <form name="/xyz2">        (action path used by modifyuser.jsp)
      <field name="id" depends="required, minlength">
        ------
      </field>
      <field name="pass" depends="required, maxlength">
        ------
      </field>
    </form>
```

Validator rules Configuration for DeleteUser.jsp

```
    <form name="/xyz3">        (action path used by DeleteUser.jsp)
      <field name="id" depends="required, mask">
      </field>
```

```
</form>
</formset>
</form-validation>
```

NOTE:- This solution1 is not suitable when multiple formpages are using single FormBean and single Actionclass.

for this use solution2

Solution2 (When multiple Formpages are using single FormBean and single Actionclass)

Step-I:- Take FormBean type as ValidatorActionForm (or) DynaValidatorActionForm

Step-II:- Configure one Action class for multiple times with different Actionpaths

Step-III:- make multiple Formpages using multiple Actionpaths on one ActionPath for Formpage basis

Step-IV:- Configure validator rules in validation.xml separately and differently for each Formpage based on the Actionpath used by each Formpage.

struts-config.xml

```
<struts-config>
  <form-beans>
    <form-bean name="rf1" type="org.apache.struts.validator.DynaValidatorActionForm">
      ------
      ------
    </form-bean>
  </form-beans>

  <action-mappings>
```

One Action class is Configured for multiple times with different Actionpaths

```
    <action path="/xyz1" type="XYZAction" name="rf1">
      ------
    </action>
    <action path="/xyz2" type="XYZAction" name="rf1">
      ------
    </action>
    <action path="/xyz3" type="XYZAction" name="rf1">
      ------
    </action>
  </action-mappings>
</struts-config>
```

FormPages

Same as above.

Validation.xml

Same as above.

NOTE:- We can configure one Actionclass for multiple times with different ActionPaths but we can't configure one FormBean class for multiple times with different logical names.

→ For Part-II of miniproject that deals with formvalidations by using DynaValidatorActionForm class refer the project given in pageno: 38 to 43 pages as application (6).

Q:- can you explain different types of Important FormBeans in Struts environment?

→ ActionForm, DynaActionForm classes support only programatic Form validations.

→ ValidatorForm, DynaValidatorForm classes can work with both programatic & Declarative (validator plugin) Form Validations. But these validations must be configured by using FormBean logical name.

→ ValidatorActionForm, DynaValidatorActionForm classes support both programatic and declarative FormValidations but we need to configure these Validations based on the action path of Struts Action classes.

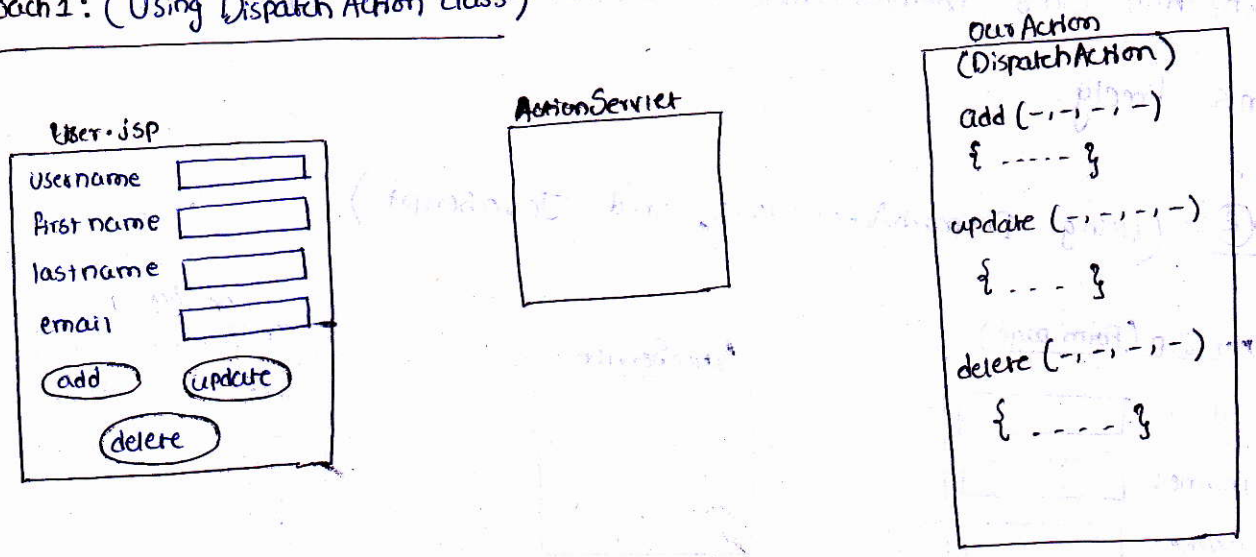
LookupDispatchAction :- LookupDispatchAction is the sub class of DispatchAction class having only one abstract method that is

```
protected Map getKeyMethodMap()
```

This LookupDispatchAction class is very useful to handle the request processing logics when Formpage contains multiple submit buttons.

To handle the Situation of Formpage containing multiple submit buttons there are three approaches.

Approach 1: (Using DispatchAction class)



NOTE:- When submit button is taken with name then the submit button caption will goto server as requestParameter Value.

Eg:- <html:submit property="s1" value="send" />.

s1 = send goes to server as request param value.

Step-I) take multiple submit buttons in formpage having different captions and same name.

```
<html:submit property="s1" value="add" />
<html:submit property="s1" value="update" />
<html:submit property="s1" value="delete" />
```

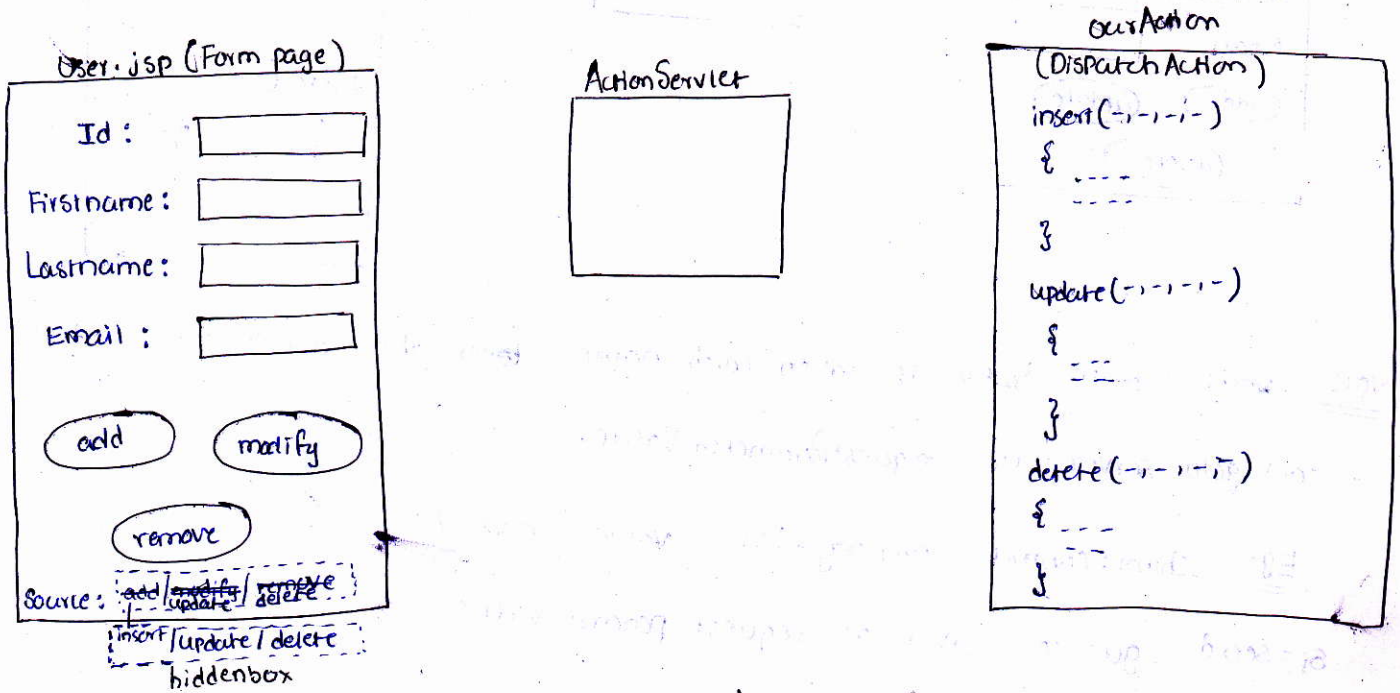
Step-II) Develop our DispatchAction class having submit buttons captions as user-defined method names (ref diagram)

Step-III) Configure our DispatchAction class in struts configure file having the name given multiple submit buttons as additional request param value.

```
<action path="/Controler1" type="ourAction" name="rf1" parameter="s1">
  ------
  ------
</action>
```

In this approach submit button captions are tightly coupled with DispatchAction class method names. So, we can't change the submit button captions freely.

Approach ② :- (using DispatchAction class, and Javascript)



Step ① :- take multiple submit buttons in Formpage having different captions.

Step ② :- Develop our DispatchAction class having different user defined methods

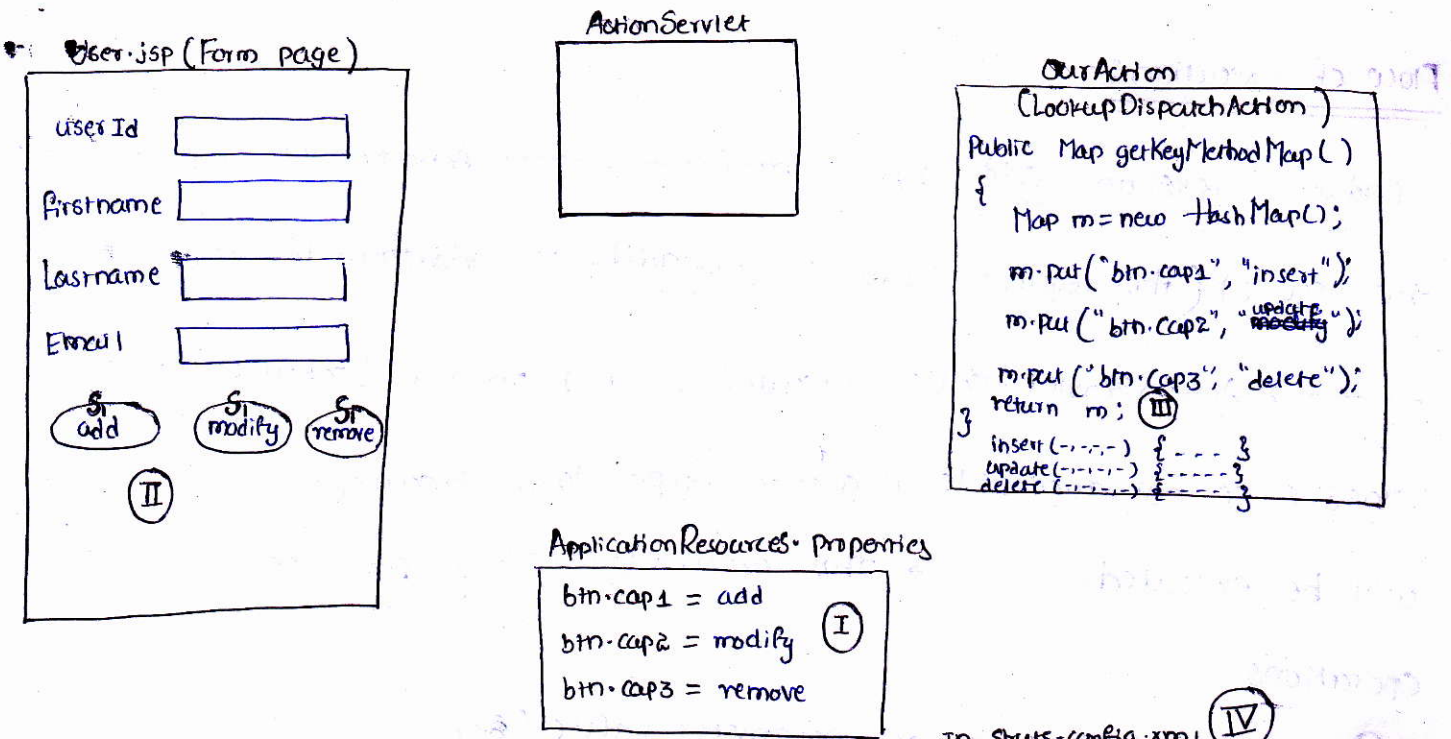
Step ③ :- Place hidden box in Formpage and generate the methodname of our DispatchAction class as dynamic value to that hiddenbox based on the submit button that is clicked (use javascript for this)

Step ④ :- Configure DispatchAction class in our Struts Configuration file having the hidden box name as additional request parameter value.

```
<action path="/xyz" type="OurAction" name="rf" parameter="source">
    ------
    ------
</action>
```

In this approach ② the submit button captions are not tightly coupled with our DispatchAction class method names but if the Javascript execution is disable through browser settings the application will not work in this approach.

Approach ③ (using LookupDispatchAction class)



In Struts-config.xml IV

```
<action path="/xyz" type="OurAction" name="rf" parameter="s1" />
```

Step Ⅰ :- prepare properties file having submit button captions

Step Ⅱ :- add multiple submit buttons to Formpage having same name, different captions gathered from properties file

s1

add, modify, remove

Step Ⅲ :- Develop our LookupDispatchAction class having multiple user-defined methods. and also implement getKeyMethodMap() method as shown in the diagram.

NOTE:- The Map object returned by getKeyMethodMap() method contains elements having keys of the submit button captions as keys contains and method names of our LookupDispatchAction class as Values.
↳ in properties file

Step-(IV):- Configure our LookupDispatchAction class in struts configuration file having the name of submit buttons as additional RequestParameter Value.

In struts-config.xml

```
<action path="/xyz" type="OurAction" name="f" parameter="s1">
    ----
</action>
```

Flow of Execution:-

End user clicks on add submit button ⟶ ActionServlet traps and takes the request (this request contains s_1=add as additional RequestParam value) ⟶ ActionServlet calls execute(-,-,-,-) on our LookupDispatchAction class. since not available it goes to SuperClass execute(-,-,-,-) method will be executed. ⟶ This execute(-,-,-,-) performs following operations

(a) reads additional request parameter value ("add" from "s_1")
(b) performs reverseLookup operation using add value and gets "btn.cap1" key from properties file
(c) calls getKeyMethodMap() method and receives HashMap object.
(d) in the received HashMap object "btn.cap1" key related value will be gatered (nothing but "insert")
(e) calls insert(-,-,-,-) of our LookupDispatchAction class to process the request.

NOTE 1:- getting value from properties file by submitting its key is called "Lookup operation".

getting key from properties file by submiting its value is called "reverse Lookup operation".

→ For example cappn on LookupDispatchAction refer cappn of the page no of (44)-(49)

26/12/2012

(*) Procedure to develop application (7) of the booklet by using My-Eclipse IDE.

Step-I:- Keep the DataBase table ready in Oracle DB slw. (Employee_info)

SQL> create table employee_info (userid varchar2(10), firstname varchar2(20), lastname varchar2(20), address varchar2(20));

Step-II:- Create web project in MyEclipse IDE

File → New → WebProject → projectName: LDApp → finish

Step-III:- Add Struts Capabalities to the project.

Right click on project → MyEclipse → add struts capabilities → ⊙Struts 1.3 → ActionServletname action → Basepackage for new classes: [ ] →

Default Application resources: ApplicationResources → finish
(properties file)

Step-IV:- Keep following entries in properties file.

ApplicationResources.properties

btn.cap1 = add
btn.cap2 = modify
btn.cap3 = remove

Step-V:- Develop DBConnection class

File → new → class → com.sathya.connmanager → class Name DBConnection → finish

refer DBConnection.java of page no: (48) & (49)

Step-VI :- add OJDBC14.jar file to the Build path of the project

Right click on project → Build path → add external archives → Browse & select ojdbc14.jar.

Step-VII :- add FormBean, Actionclass, Formpage to the project.

Right click on project → Other → MyEclipse → webstructs → Struts 1.3 → Struts 1.3 Form, Action & jsp → next → Name: [ef] (formbean logical name) → superclass: [org.apache.ActionForm] → Formtype: [EmpForm] → [FormProperties] tab → [add] userid, Firstname, lastname, address property → [Methods] tab → Deselect all checkbox → [jsp] tab → [✓] create jsp form?

New jsp path: [/User.jsp] → next → path: [/emp] → type: [EmpAction] → [Form] tab Name: [ef] (formbean logical name) → [Parameter] tab Parameter: [function] → [method] tab [✓] public AF execute(-,-,-) → [Forwards] tab → (to specify result pages)

Name: success
path: /User.jsp
[add] [close]

Name: failure
path: /failure.jsp
[add] [close]

Step-VIII :- add Multiple Submit buttons to the formpage collecting the captions from properties file

refer User.jsp of page No: (44)

Step: IX :- Develop the following code in Action class.

refer EmpAction.java of page No: (46)

Step: X :- add failure.jsp to the webroot folder of the project

File → new → jsp → Filename: [failure.jsp] → finish.

<%= request.getAttribute("operation") %> is failed...

Step-XI :- add following code in User.jsp

<%= request.getAttribute("operation") %> is successful...

Step-XII :- Configure any Server with MyEclipse IDE.

rightclick on project ⟶ ~~MyEclipse~~ run as ⟶ MyEclipse Server application

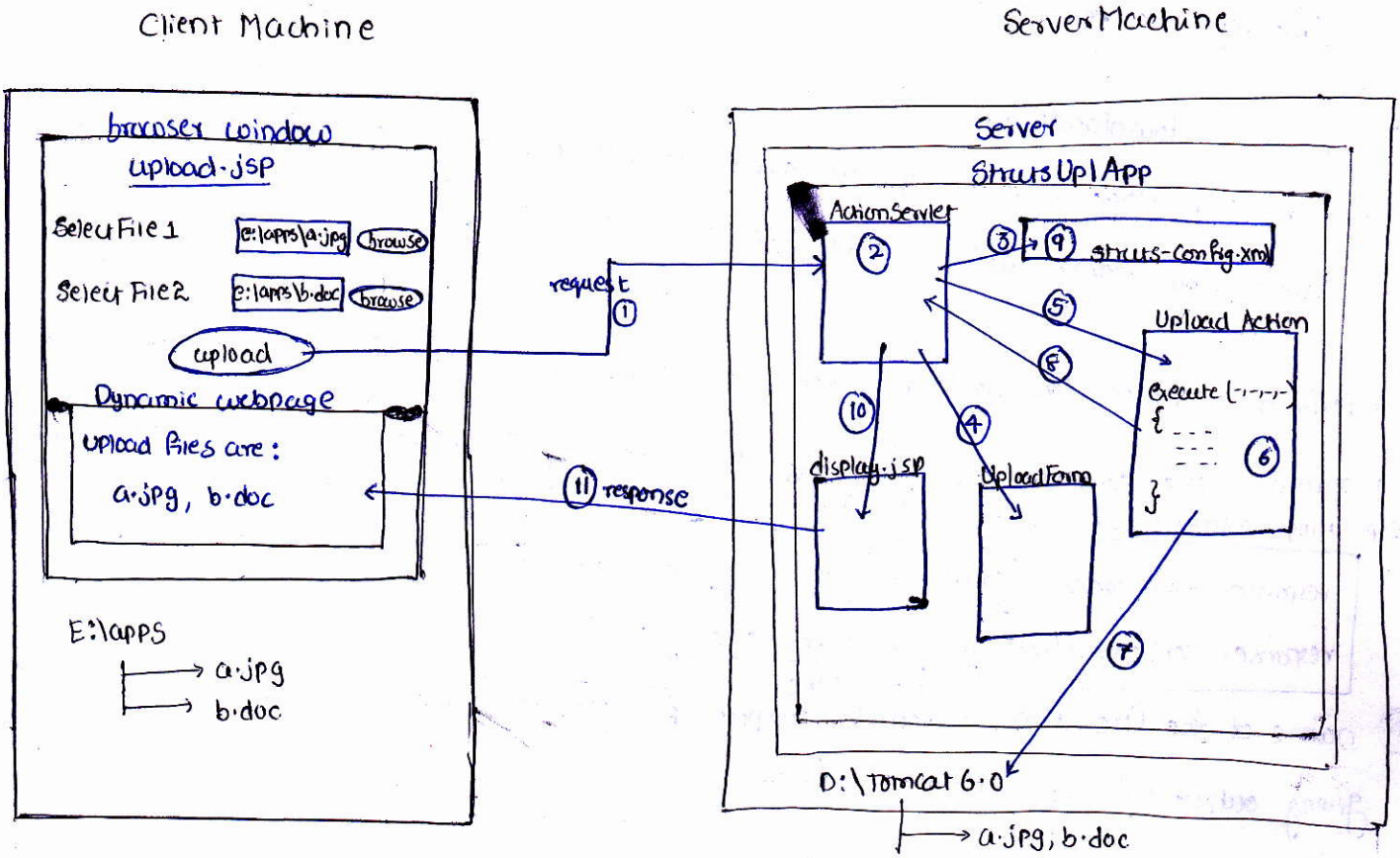
File uploading and Downloading :- 27/12/2012

→ The process of sending file from client machine file system to Server machine file system is called fileuploading and reverse is called Filedownloading

→ Struts gives built-in support for Fileuploading.

→ while developing Jobportal, Matrymony applications FileUploading and downloading operations are quite important.

→ All files and folders of a Computer together is called FileSystem.



Procedure to develop the FileUploading appn using Struts :-

Step-I :- Design Formpage having FileUploading components to select the files from client machine Filesystem.

Select File1 : <html:file property="file1" />
Select File2 : <html:file property="file2" />

Step-II:- Design FormBean class having the properties of type org.apache.struts.upload.FormFile.

FormFile type properties are "streams" representing the Files that are uploaded.

Step-III:- Develop the streams based logic in the execute(-,-,-,-) of Struts Action class to write the uploaded files to Server machine File system.

Step-IV:- Develop the remaining resources of the application in Normal manner.

For example application on File uploading refer application (11) of Page No's: (93) - (95)

File downloading:-

We can see two types of file downloading.

(1) response downloading
(making the response of webresource prg as downloadable file)

(2) resource downloading
(making other resources of webappn as downloadable files)
Eg:- downloading audio, vedio, zip and other files.

To perform response downloading place the following 2 lines of code in any webresource program like servelet, jsp programs to make their output as downloadable files.

in jsp using scriptlets (<% %>) and write this code in scriptlets, in servlet this code is written in service(-,-) (or) doXxx(-,-)

```
response.addHeader("Content-Disposition", "attachment-filename=Title1.xls");   (*1)
response.setContentType("application/ms-excel");
```

(response header; mimetype)

(*1) name of the file that contains output of current webresource program, while giving output as downloadable file.

NOTE:- The above two lines of code can be used in any webresource program that gives access to request, response objects.

To perform resource downloading in struts appns we must work with Action class that extends from org.apache.struts.action.DownloadAction.

Download Action class of Struts API is an abstract class containing two inner classes and one inner interface. they are

① FileStreamInfo (inner class)

② ResourceStreamInfo (inner class)

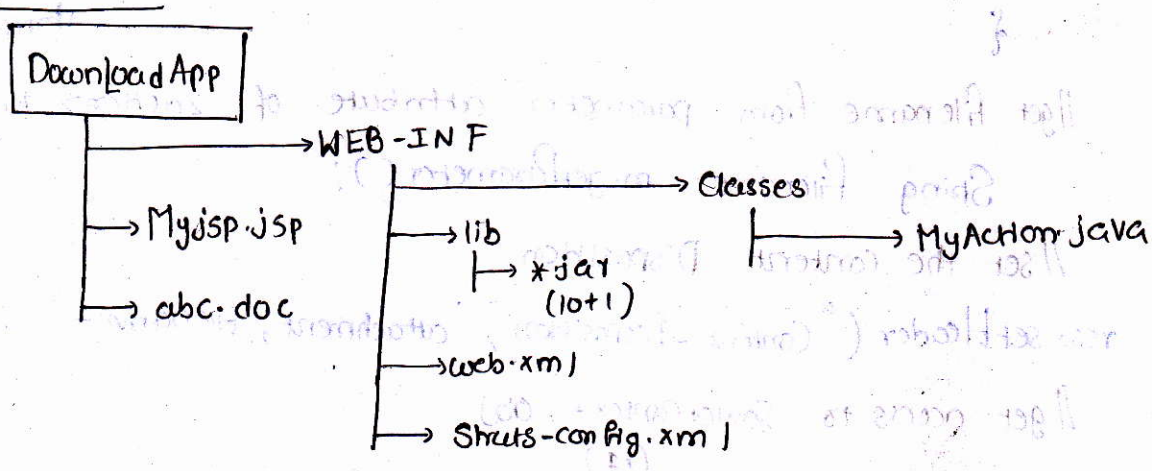
③ StreamInfo (inner interface)

the ①, ② inner classes implementing the ③ inner interface (StreamInfo)

28/12/2012

(*) Example appn on Resource download in the support of Download Action class.

Directory structure:-



Jarfiles in CLASSPATH:-

Servlet-api.jar, Struts-core-1.3.8.jar, Struts-extras-

Myjsp.jsp :-

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<html:link action="/dpath">Download Word Doc</html:link>
```

↳ action path of DownLoadAction class

Web.xml:-

Same as other Struts appn.

Struts-config.xml:-

```
<struts-config>
    <action-mappings>
        <action path="/dpath" type="MyAction" parameter="/abc.doc" />
    </action-mappings>
</struts-config>
```

↳ The filename that we want to download

```
// MyAction.java
import   java.io.*;
import   javax.servlet.*;
"        "      "   .http.*;
import   org.apache.struts.action.*;
"        "      "   "   .actions.*;

public   class   MyAction   extends   DownLoadAction
{
protected   StreamInfo   getStreamInfo(ActionMapping m, ActionForm f,
                                       HttpServletRequest req, HttpServletResponse res)
                                                                    throws Exception
{
//get filename from parameter attribute of <action> tag
   String filename = m.getParameter();
//set the content Disposition
res.setHeader("Content-Disposition", "attachment; filename="+filename);
//get access to ServletContext obj
   ServletContext   ctx = servlet.getServletContext();
//return stream object pointing to the file to be downloaded
return new ResourceStreamInfo("application/ms-word", ctx, filename);
}
}
```

(*1) → servlet is member variable of org.apache.struts.action.Action class holding the reference of ActionServlet class object. This is visible in every Struts Action class.

Switch Action :-

→ every Struts application contains one default module (module without name)

while developing large scale Struts application it is recomended to create multiple modules in that Struts appn having multiple Struts Configuration files on one per module basis.

→ If we work with single Struts configuration file there is a possibility of getting naming colashes to overcome this, work with multiple Struts Configuration files by creating multiple modules in Struts appn.

→ The explicitly created modules of Struts appn are called named modules

If Struts appn is having two named modules then we can say total 3 modules are there in that Struts appn. (1 default, 2 Named)

→ To specify the Struts Configuration files of multiple modules we need to write the following entries in web.xml file while configuring ActionServlet

in Web.xml

```
<servlet>
    <servlet-name> action </s-n>
    <servlet-class> ActionServlet </s-c>

    <init-param>
        <parameter-name> config </parameter-name>
        <parameter-value> /WEB-INF/ struts-config.xml </p-v>
    </init-param>

    <init-param>
        <p-n> config/mod1 </p-n>
        <p-v> /WEB-INF/ struts-config-student.xml </p-v>
    </init-param>

    <init-param>
        <p-n> config/faculty </p-n>
        <p-v> /WEB-INF/ struts-config-faculty.xml </p-v>
    </init-param>
</servlet>
```

Struts config file of default module

Struts config file of mod1 module

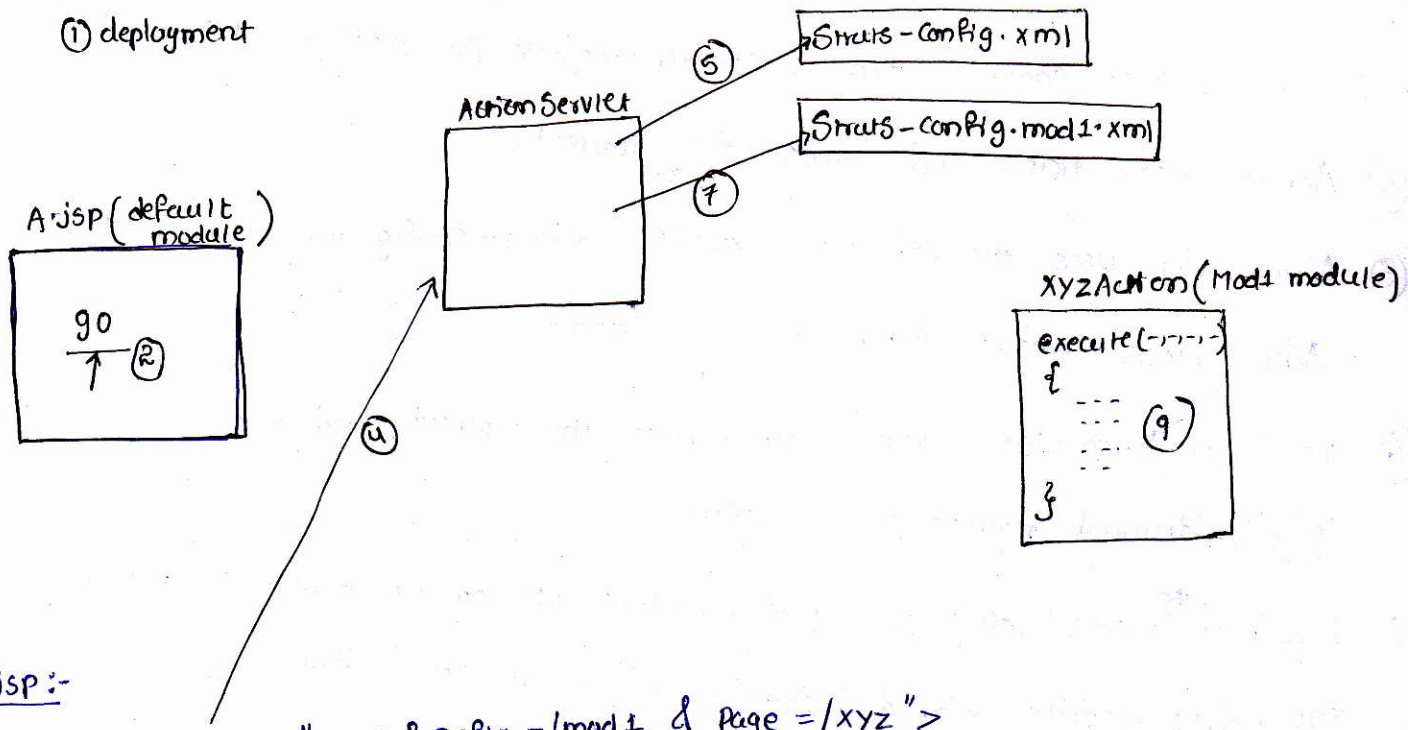
Struts config file of faculty module

→ The built in Action class org.apache.struts.actions.SwitchAction is given to navigate the Controle blw two different resources of two different modules. SwitchAction should always be configured in source module related struts configuration file. and we should get request having two additional request parameter values they are

1) Prefix :- holds the target module name.

2) Page :- holds the target resource identification name.

Navigation blw default module resource and NamedModuleResource :-



A.jsp :-

```
<html:link  action="switch?prefix=/mod1 & Page=/xyz">        ③
go
</html:link>
```

Struts-config.xml :-

```
<action  path="/switch"  type="org.apache.struts.actions.SwitchAction" />        ⑥
```

Struts-config-mod1.xml :-

```
<action  path="/xyz"  type="XyzAction" />        ⑧
```

NOTE :- while working with SwitchAction class the resource of source module gives request to SwitchAction and this SwitchAction navigates the Controle to Specific resource of destination module. For this SwitchAction ~~Action~~ should get request having target modulename and target resource name has prefix, page additional request parameter values.

29-12-2012

W.r.to the diagram

(1) programer deploges struts application in webserver or application server

In this process the instantiation and initialization of ActionServlet takes place.

ActionServlet also reads various configue init-parameter values to recogrize module names and their struts configuration filenames in struts application.

(2)&(3) Enduser Clicks on "go" hyperlink of A.jsp that belongs to default module.

In this process based on URL kept in the action attribute of <html:link> tag request will be generated to struts application.

This request contains two additional request parameters prefix, pairs

(4) ActionServlet traps and takes the request.

(5) ActionServlet uses the default module "struts-config.xml" file to locate Action class whose Action path is "/switch"

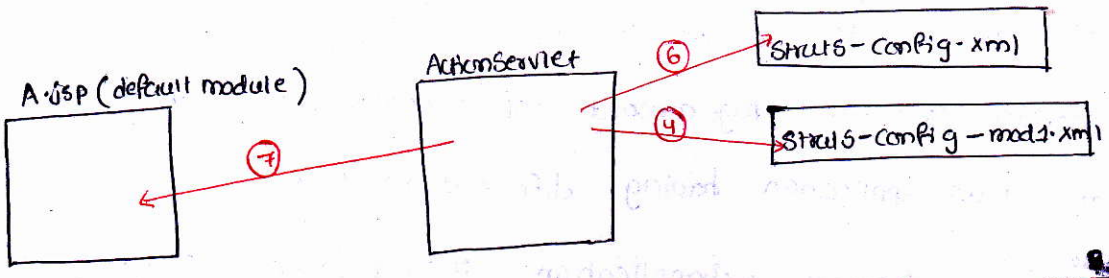
(6) The Switch Action class traps and takes the request and also reads prefix, page additional request param values.

(7) Based on the mod1 value of "prefix" additional request parameter controle goes to the mod1 module related "struts-config-mod1.xml" file

(8) Based on page=XYZ the XYZAction class of whose action path is "/XYZ" will be located in struts-config-mod1.xml file

(9) ActionServlet calls execute(-,-,-,-) method on XYZAction class object.

Navigation between Named module Resource and Default Module Resource



① Deployement of Struts app

B.jsp
(mod1 module)
Go ②

Struts-config-mod1.xml

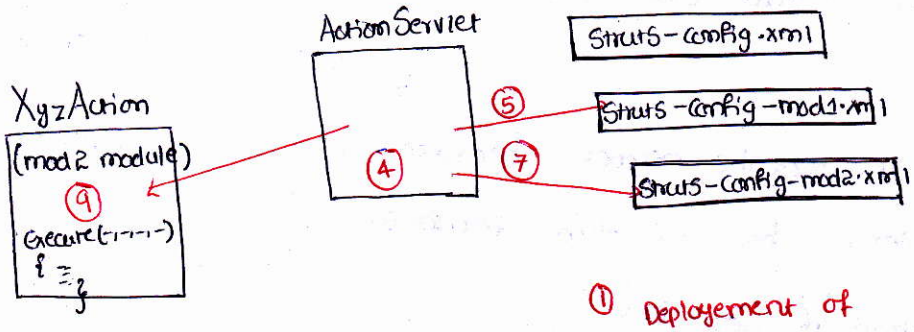
<action path="/switch" type="org.apache.struts.actions.SwitchAction" />
⑤

B.jsp

<html:link action="switch?prefix=&page=/A.jsp">
③
Go
</html:link>

prefix =
↳ no value i.e. means default module value it takes

Navigation between Named module Resource and Named module Resource



A.jsp (mod1 module)
Go ②

① Deployement of Struts app

Struts-config-mod1.xml

<action path="/switch" type="org.apache.struts.actions.SwitchAction" />
⑥

Struts-config-mod2.xml

<action path="/xyz" type="XyzAction" />
⑧

A.jsp

③
Go

→ In every module related Struts configuration file the FormBean, Action classes, Jsp programs of that module will be configured

→ If Struts application contains huge amount of resources, then think about developing that Struts application having different modules

→ Each module is not a separate webapplication, it is a logical partition in the web appln. The web appln that represents college operations contains the following module.

① Faculty module.
② Student module.
③ Academics module.
④ Library management module.

→ we can send only Serializable objects over the network. object becomes Serializable object only when the class of that object implements java.io.Serializable interface.
(Serializable ← marker interface)

→ we can't send ResultSet object over the network because it is not Serializable object. To solve this problem there are two solutions.

① Use RowSets instead of ResultSets

→ RowSets are Serialized objects by default
→ Very few jdbc drivers are supported Rowsets.

② Copy the records of ResultSet obj to Collection Framework DataStructure and send that DataStructure over the network

→ All Collection Framework DataStructures are Serializable objects.

Solution ② is recommended to use

→ If you are looking to perform only read operations on the data of Collection Framework DataStructure then use non-synchronized datastructures for better performance Eg: ArrayList, Hashmap and etc...

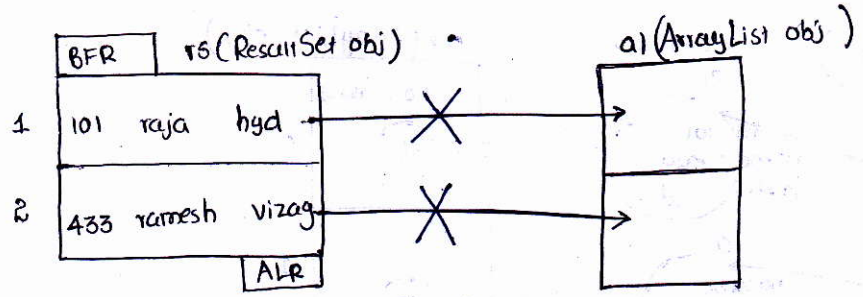
→ If you are looking to perform both read and write operations on the data of

collection framework datastructure then use Synchronized DataStructure for acheiving Thread safety.

Eg:- Hashtable, vector and etc..

→ to send collection frame work DataStructure over the network the objects adding to the elements of DataStructure must be taken as Serializable objects. otherwise we can't send that DataStructure over the network.

→ we can't copy each record of ResultSet to each element of ArrayList because each record of ResultSet may contain multiple objects but each element of ArrayList allows only one object. But this operation can be done indirectly with the support of D.T.O class/ V.O class.



X Indicates that we can't copy records of ResultSet obj directly to the elements ArrayList obj

30/12/2012

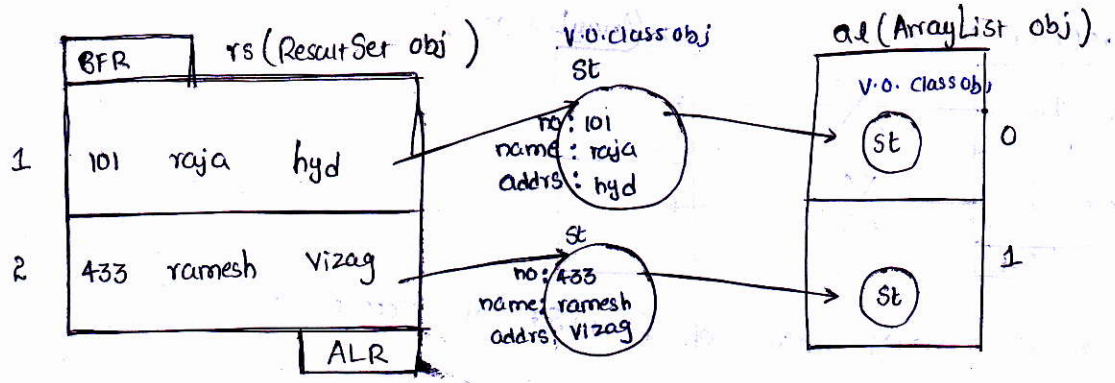
To solve the above problem copy each record of ResultSet to One object of user defined java class and add that Object to each element of arrayList. This user defined java class is javabean is called D.T.O class (or) V.O class.

StudentBean.java (D.T.O class | V.O class)

```
public class StudentBean implements java.io.Serializable
{
    int no;
    String name, addrs;
    //write getXxx(), setXxx(-) methods
    .....
    .....
}
```

Logic to transfer records of ResultSet obj to ArrayList elements :-

```
ResultSet rs = st.executeQuery("select * from student");
ArrayList al = new ArrayList();
while(rs.next())
{ //copy each record to one new StudentBean class obj
  StudentBean st = new StudentBean();
   st.SetNo(rs.getInt(1));
   st.setName(rs.getString(2));
   st.setAddrs(rs.getString(3));
  //add StudentBean class obj to ArrayList element
     al.add(st);
} //while
```



Q:: Where did you use Javabean in your project?

Ans:- ① as D.T.O (or) V.O class while copying the records of ResultSet to ArrayList elements.

② as persistent class in hibernate applications.

③ as FormBean class in struts 1.x applications

④ as ModelLayer resource in MVC1, MVC2 applications

⑤ as springBean in Spring applications.

⑥ as V.O class to store multiple values of FormPage in single object during SessionTracking.

and etc....

→ while gathering jdbc properties from text properties file from we need java.util.Properties class

Q:- Where did you use collections in your project?

→ To send ResultSet records over the network we copy them to ArrayList.

→ To maintain mail properties in java mail programing we need java.util.Properties class

→ To maintain Jndi properties we need Map DataStructure.

→ To maintain huge amount of data in sessionTracking we add DataStructures to Session attributes.

→ To maintain huge amount of data and dynamically growable data we take CollectionFramework DataStructures and etc....

* For Example appn on SwitchAction and the above discuss D.T.O class refer page NO's (63) to (69) of the booklet.

31/12/2012

→ even though network is not there it is not recomended to send ResultSet obj from Source Layer to destination Layer directly. Because writing Jdbc code in Destination Layer to receive ResultSet obj is not recomended process.

To overcome this problems Copy the Records of ResultSet to CollectionFramework DataStructure and send that DataStructure from Source Layer to Destination Layer directly.

I18N (Internationalization)

→ Making our appn working for different locales by rendering numbers, dates, Currency Symbols, labels with locale specific format. is called enabling internationalization on the application.

Locale means Country + Language

Eg:- en-US
fr-FR
de-DE and etc....

→ by enabling i18n on the application, websites we can make more an endusers Clients, utilizing our projects.

→ i18n is no way related with Bussiness Logic of the appn. it is purely related to Presentation Logic of the appn.

→ To enable i18n we need to work with multiple properties files on one per Locale basis

App.properties (Base properties file)
App_fr_CA.properties (for french as it speaks in CANADA)
App_de_DE.properties (for german as it speaks in GERMANY)
App_hi_IN.properties (for hindi as it speaks in India)

→ Generally Base property file lable Contains default property file that executes when matching property file is not found. The Base filename must be there in every locale specific because this is properties file.

→ Java supports Unicode character set in which almost all popular languages related alphabets and supplimentries are included.

→ By using Unicode editor, native2ascii tools we can get Unicode number for our regional language content.

Procedure to get Unicode numbers for the following hindi words

#(BaseFile → English)

निकाली

सुरक्षित

रूखी

रद्द

Step-I:- Install uni-code editor by getting it from www.higopi.com

Step-II:- launch uni-code editor and type the following hindi alphabets

निकाली सुरक्षित रूखी रद्द

Step-III:- copy the above hindi words to a text file → save "a1.txt"
(choose uni-code encoading while saving the file)

Letus assume the above content is copied to a1.txt.

Step-IV:- Use the Jdk supplied native2ascii tool as shown below to get Unicode numbers.

>native2ascii -encoading unicode a1.txt code1.txt

↳ contains unicode numb of the above hindi words

02/01/2013

#for french Language

These german, french words can be gathered by using google Tranlated tool.

APP_hi_IN.properties

for hindi language

str1 =

str2 =

str3 =

str4 =

we can gather these Unicode numbers by using native2ascii tool as discussed above.

NOTE:- While working with i18n the keys of multiple properties file must be taken Same.

```
//I18nApp.java
// internationalization of an application (the captions on buttons will come
// based on the locale specific properties file that is activated)

import java.awt.*;
import javax.swing.*;
import java.util.*;

public class I18nApp {
   p s v m(String[] args) {
   // create locale object based on given language, country codes.
Locale l = new Locale(args[0], args[1]);

   //Picks up properties file based on the Locale obj data
ResourceBundle r = ResourceBundle.getBundle("App", l);

   JFrame jf = new JFrame();
Container cp = jf.getContentPane();

//Create buttons by getting labels from activated properties file.
JButton b1 = new JButton(r.getString("str1"));
JButton b2 = new JButton(r.getString("str2"));
JButton b3 = new JButton(r.getString("str3"));
JButton b4 = new JButton(r.getString("str4"));

cp.setLayout(new FlowLayout());
cp.add(b1);
cp.add(b2);
cp.add(b3);
cp.add(b4);

jf.pack();
jf.setVisible(true);
} //main
} //class
```

/> javac I18nApp.java
/> javac I18nApp fr CA (APP_fr_CA.properties file will be activated)
/> javac I18nApp hi IN (APP_hi_IN.properties file will be activated)
/> javac I18nApp x y (in this case no property file is given then Base property file will be activated)

Struts gives built-in support for I18n. with also internally uses locale, ResourceBundle classes for this.

we need to perform following operations to enable I18n on Struts application.

Step-I:- prepare multiple locale specific properties file and keep them in WEB-INF/classes folder.

Step-II:- Configure these properties file in Struts Configuration file.

Step-III:- use <bean:message> tag in Jsp program to gather presentation labels from properties file.

Step-IV:- Deployeed Struts appn in any Server.

Step-V:- using browser Settings choose one Language & Send request to Struts appn.

Note:- The Language choosen in Step-V becomes accept-language request header value when Struts appn takes the request it internally uses this accept-language header value to pickup the locale specific appropriate properties file.

Eg:- if French Language (fr) is choosen through browser Settings then Struts appn looks to use base filename_fr.properties file.

For example appn on Struts based I18n refer page no: 91 Appn (10).

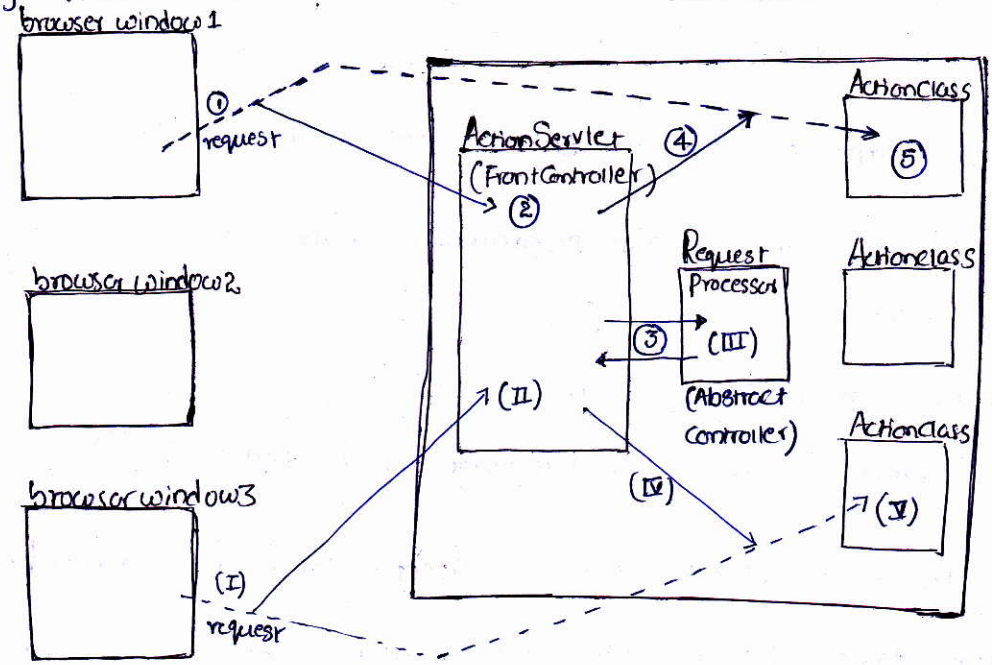
RequestProcessor

03/01/2012

Abstract Controller :- It is a FrontController for servlet / jsp using this we can customize the logics of FrontController is called AbstractController class.

→ In Struts 1.x org.apache.struts.action.RequestProcessor class is abstract Controller class for the frontController ActionServlet.

→ RequestProcessor class takes care of all Request Processing operations on behalf of ActionServlet. The support of process(), processXxx() methods. These operations are like creare/locating FormBean class objs, Actionclass objs, Calling validate(), execaure() methods and ete.



→ ActionServlet creates objects per RequestProcessor class on one per Struts module basis and total RequestProcessing operations and response generation operations related to each module will be taken care by calling 16+ processXxx() methods on each module related RequestProcessor class object.

16+ → 19 in Struts 1.x

→ Every request given to Struts appn under goes 16+ RequestProcessing operation by calling 16+ ProcessXxx() methods of RequestProcessor class.

These methods are like ProcessPopulate(), processValidate(), processMultipath and ete...

Template Method Design patterns :-

Problem :- **task1 :-**

```
a();
b();
c();
d();
```

Here we need to call Three methods by remembering their sequence if invocation to complete task

Solution :- Design Template method.

```
Public void xyz()
{
    a();
    b();
    c();
    d();
}
task1: xyz();
```

The method i.e called multiple other methods that in a sequence to complete a task is called template method due to this programmer just need to remember and call only one method and there is no need of remembering and calling multiple other methods in a sequence to complete the task.

→ Process() of RequestProcessor class internally calls 16+ processXxx() methods in a sequence. so, this method is called Template method. Instead of calling the 16+ processXxx() methods in a sequence on RequestProcessor class object the ActionServlet calls only process() method on RequestProcessor class object.

→ Related information on RequestProcessor class and its process, processXxx methods refer pageno: 70 to 73.

→ To provide additional instructions, inputs to various processXxx() methods of predefined RequestProcessor class use <controller> tag of struts configuration file as shown in pageno's 71 & 72

→ programmer can develop user-defined Abstract controller class for ActionServlet to customize RequestProcessing operations. It is recommended to develop user defined abstract controller class by extending from pre-defined RequestProcessor class.

The User-defined abstract controller class is also called as Custom Request Processor class.

Procedure to add CustomRequestProcessor class for mod1 module of SwitchAction application to allow only InternetExplorer generated request to ActionClasses of mod1 module.

Step-I :- keep switchAction appn ready

Step-II :- develop the CustomRequestProcess class in WEB-INF/classes folder as shown below.

```
// CustRp.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import org.apache.struts.action.*;
public class CustRp extends RequestProcessor
 {
  public boolean processPreprocess(HttpServletRequest req, HttpServletResponse
                                                        res)
  {
try
 {
  // get browser coindow name of request
  String brname = req.getHeader("user-agent");
  if(brname.indexOf("MSIE") != -1) // block the request coming from other than IE
   {
    return true;
   }
  else
   {
    RequestDispatcher rd = req.getRequestDispatcher("/err.jsp");
    rd.forward(req,res);
    return false;
   }
 //try
   catch(Exception e)
    { p.printStackTrace(); return false;
    }
 }// ProcessPreprocess()
}// class
```

I> javac CustRp.java

Step-III :- add error.jsp in web appn.

 <Font color=red> Request must be given from InternetExplorer </Font>

Step-IV :- configure the above RequestProcesser class in Mod1 module related Struts Configuration.

in Struts-config-student.xml after </action-mappings>

```
</action-mappings>
  <controller>
    <set-property property="processorClass" value="CustRp" />
  </controller>
</struts-config>
```

Step-V :- Test the appn in regular manner.

We can configure CustomRequestProcessor class on per module basis not on per struts appn basis.

NOTE :- The above appn is using pre-defined RequestProcessor class as abstract controller for default, faculty module. (by creating two objs for RequestProcessor class) and using CustRp class as Abstract controller for mod1 module (by creating one obj for CustRp class).

04/01/2013.

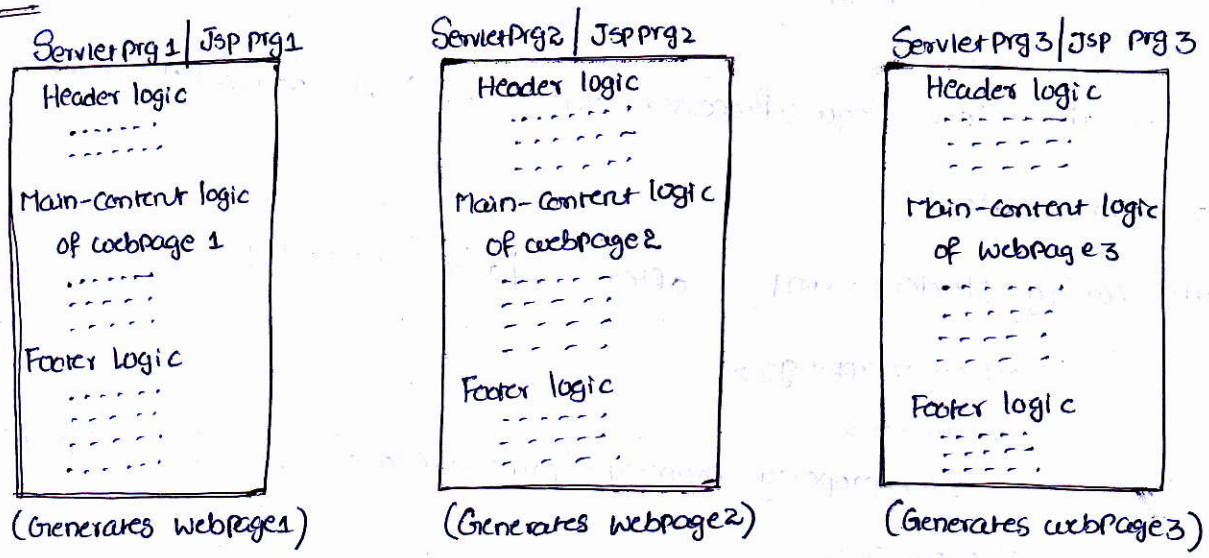
→ while developing CustomRequestProcessor class we generally override process() preProcess() methods to place programmer choice. Because it is the empty method of predefined RequestProcessor class.

→ To check weather user is there in the session or not to show either Requested page or Login page we take the support of filters in Servlet programing. To do that work in Struts environment we take the support of CustomRequestProcess class

→ for realworld scenario placed CustomRequestProcessed development refer page no's (74) & (75) of the booklet.

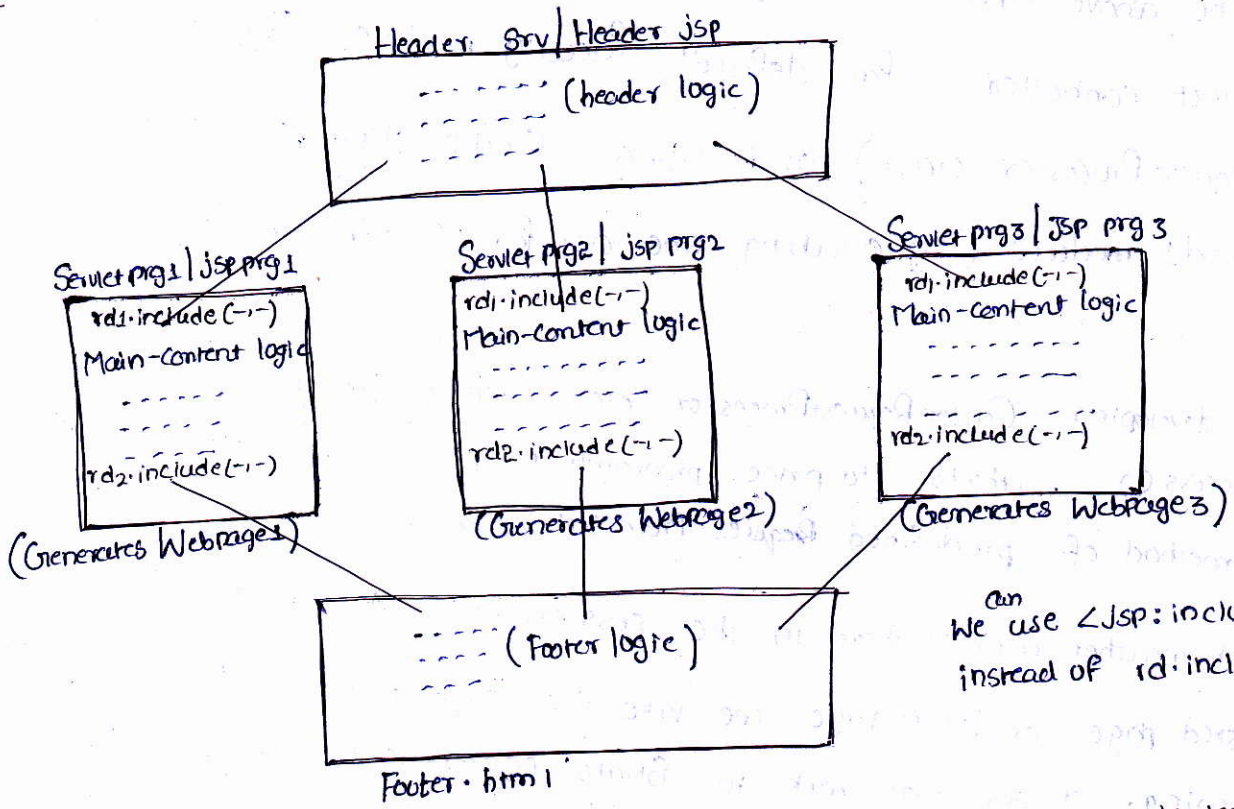
Working with Tiles-plugin (or) Working with Tiles Framework :-

Problem:-



All webpages of website contains same Header, Footer content. But they contain different main content. In the above 3 servlet program | Jsp program header and Footer logics are placed in every program even though they are same. This indicates header, Footer logics are not reusable logics.

Solution:- work with Composite-view Design pattern.

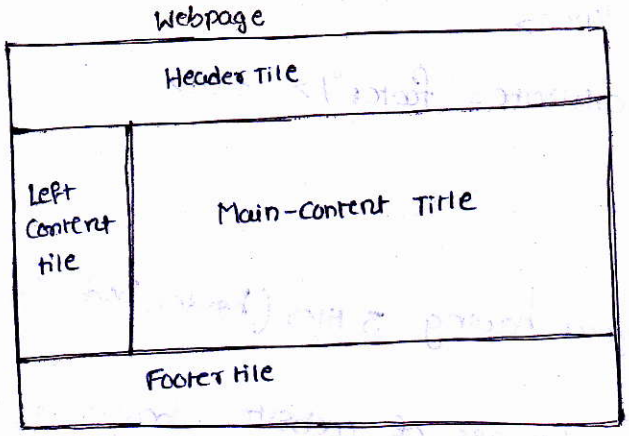


→ Composite View Design Pattern makes the programer to place common logics with Separate web resource programs and uses those logics for multiple times in other main web resource programs as shown in the above diagram.

Tiles:-

A Tile is logical partition of the webpage representing certain content of the webpage

One webpage can have any no.of Tiles.



→ The Tiles-plugin of Struts is given ~~to develop~~ to integrate Composite view DesignPatter by satisfying MVC2 rule and having Layout Controller.

→ Designing ~~the~~ webpages based on Layout page and reflecting the modifications to webpages based on the modifications done in Layout page is called Layout Controller.

→ In Struts 1.x environment the Tiles plug-in related resources and logics are available <Struts-home>\lib\struts-tiles-1.3.8.jar

Procedure to work with Tiles-plugin in our Struts appn :-

Step-I:- Configure Tiles plug-in in Struts Configuration file.

```
<plug-in eclassName = "org.apache.struts.tiles.TilesPlugin">
    <set-property property = "definitions-config" value = "/WEB-INF/tiles-defs.xml"/>
                                                          └→ Tiles Configuration file
    <set-property property = "moduleAware" value = "true" />
                              └→ makes the Tiles plug-in to recognize modules.
</plug-in>
```

Step-II:- Design Layout page having Tiles.

```
<%@ taglib uri = "http://struts.apache.org/tags-tiles" prefix = "tiles" %>
<table border = "0" width = "100%" height = "100%">
  <tr width = "100%" height = "20%">
    <td> <tiles:insert attribute = "header" /> </td> <!-- header Tile -->
  </tr>
```

```
<tr width="100%" height="70%">
    <td> <tiles:insert attribute="body" /> </td>  <!--body file-->
</tr>
<tr width="100%" height="10%">
    <td> <tiles:insert attribute="footer" /> </td>  <!-- footer file-->
</tr>
</table>
```

this Layout.jsp is the layout having 3 tiles (header, body, footer)

Step-III :- Design the no.of webpages of website based on this Layout (Let us assume 3 pages)

Step-IV :- Develop tile configuration file having tile definition specifying the values of tiles for each webpage.

(refer xml code of page no : (81))

Step-V :- Create the Action class in regular manner.

```
public class MyAction extends Action
{
  Public ActionForward execute(-,-,-,-)
  {
    ----
    if(condition1)
        mapping.findForward("res1");
    if(condition2)
        mapping.findForward("res2");
    if(condition3)
        mapping.findForward("res3");
  }
}
```

Step-VI :- Configure Action class in Struts configuration file specifying file defination files as result page.

Struts-config.xml

```
<action path="/XYZ" type="MyAction" >
    <forward name="res1" path="adef"/>
```

<forward name="res2" path="bdef" />
<forward name="res3" path="cdef" />

</action>

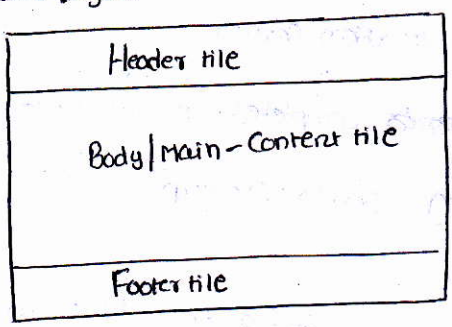
Step-VII:- Develop the remaining resources in reguler manner.

There is the possibility of designing the tile definitions of inheritence tile defination configuration to inheritance as shown in page no: (83)

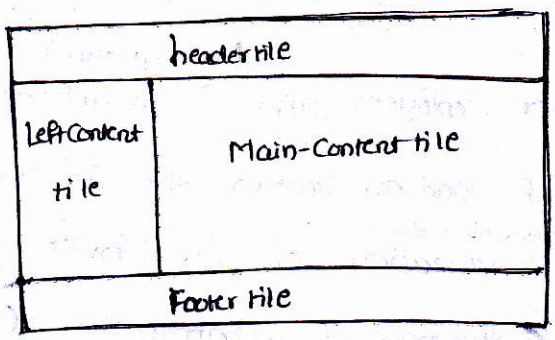
There are some populer layouts to design layout pages.

1. Classic Layout
2. Two Column Layout
3. Three Column Layout
4. Circuler Layout and etc....

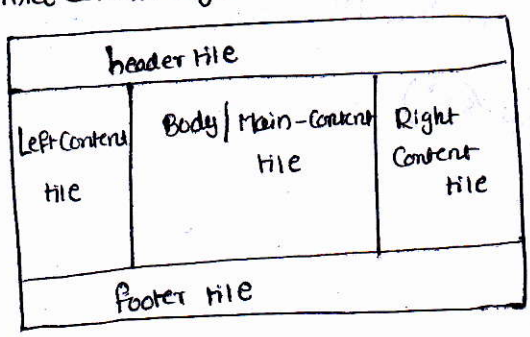
① Classic Layout



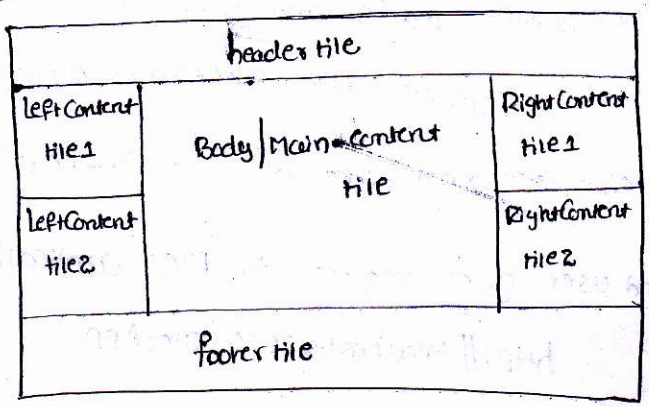
② Two Column Layout



③ Three Column Layout



④ Circuler Layout



D.A.O (Data Access Object)

The class of component that separates persistence Logic from other Logics of the appn to make persistence Logic as the flexible Logic to modify is called D.A.O.

→ This D.A.O class contains Logics establish the connection blw database slw, to release the connection with database slw and to perform various CURD Operations.

→ Logics of D.A.O class can be developed by using persistence technology like hibernate, Jdbc and etc.....

→ The multiple Action classes of Struts application either Single D.A.O class (or) multiple ~~Struts Config~~ D.A.O classes.

For tiles framework based mini project refer application (9) of the page no's (84) to (90)

The flow of execution of appn (9) given in the booklet.

NOTE:- while working with tiles to make the 1st page of website coming based on tiles definition. the struts appn must get one implicit request through welcome page. for this we can use rd.forward(-,-), (or) jsp:forward (or) response.sendRedirect(-,-) method in that welcome page as shown in index.jsp of appn (9).

(A) programmer deployees appn (9) (TilesApp) in webServer or application Server.

(B) because of load-on-startup the Servlet Container ~~reads~~ completes pre-instantiation (object creation) and pre-initialization (executing method) of ActionServlet either during Server Startup and during deployement of webapplication. (ref 23)

(C) ActionServlet reads and verifies the entries of Struts-config.xml file and also recognizes the tilesplugin configuration. In this process the entries of tiles-defs.xml coill also be verified (refer (82)-(85)).

(D) End user gives request to Tiles application

http://localhost:2020/TilesApp

(E) The default welcome page index.jsp is executed & this page sends implicit request to TilesApp web application. (refer (32)) & (ref (5))

(F) ActionServlet traps and takes the implicit request given generated by index.jsp. (ref 16-24 & 26-29)

(G) ActionServlet passes the controle to <action> tag whose Action path is "/mytiles". (ref (52))

(H) Controle goes to tile defination whose name is "MyTiles". (ref linno: 425-430)

(I) The Layout page MyTilesHome.jsp will be displayed on the browser window having following Tile values.

header tile value ⟶ BaseHeader.jsp

footer tile value ⟶ BaseFooter.jsp

left-content tile value ⟶ LeftNavigation.jsp

main-content tile value ⟶ welcome.jsp

(J) EndUser clicks on insert hyperlink to generate the request (ref: 454)

(K) ActionServlet traps and takes the request (ref: 16-24 and 26-29)

(L) ActionServlet process the Controle to <action> tag whose action path is "insert."

(M) Controle goes to tile defination whose name is "insertTile" (ref: 438-440)

(N) The Layout page MyTilesHome.jsp will be displayed on the browserwindow having following Tile values.

header tile value ⟶ BaseHeader.jsp

footer tile value ⟶ BaseFooter.jsp

left-content tile value ⟶ LeftNavigation.jsp

main-content tile value ⟶ insert.jsp

(O) Insert.jsp related FormBean class object will be created internally. (ref: 42-46)

(P) EndUser submits the request from insert.jsp from page using submit button & this generates request (ref: 126 & 114)

(Q) ActionServlet traps and takes the request (ref: 16-24 and 26-29)

(R) ActionServlet copies form data to FormBean class obj (ref: 42-46)

(S) ActionServlet passes the request to insertAction class whose action path is "/insertData"

(T) ActionServlet calls execute(-,-,-,-) on insertAction class obj, (ref: 336-357)

note:- This execute(-,-,-,-) internally calls insertData(-,-,-) of DAO class to insert the record. (ref 241-263)

Sucess		failure
354	(U) (returns ActionForward obj)	356
365-67	(Uses ActionForward configurations) (V)	68-70

(W)

432-434 (Uses tile defination cfgs) 435-437

(X) The Layout page MyTilesHome.jsp will be displayed on the brocosor coindoco having the following tile values

header tile value ⟶ BaseHeader.jsp
footer tile value ⟶ BaseFooter.jsp
left-content tile value ⟶ LeftNavigation.jsp
main-content tile value ⟶ Success.jsp | failure.jsp

If multiple threads are acting on Single Object Simultaneously or concurrently there is that Object is not threadsafe object.

To acheive Thread Safety we need to cooperate with Synchronization.

ThreadSafety means allowing only one thread at a time on to the object.

→ Struts Action Class is not threadsafe by default. because when multiple requests are given to Single Struts Action class then multiple threads will be started Simultaneously or Concurrently on One object of StrutsActionclass.

→ To make StrutsActionclass as Threadsafe class place Synchronized blocks inside the execute(-,-,-,-) method.

```
public class MyAction extends Action
{
   public ActionForward execute(-,-,-,-)
   {
      Synchronized (con)
      {
         ----- //jdbc con related stmts
         ---
         ---
      }
      Synchronized (session)
      {
         ----- //session obj related stmts
         ---
         ---
      }
      ----- //other logics
      ---
      ---
   }
}
```

07/01/2013

Exception-Handling in Struts

Exception is a Run-time error. we can't see that is not occuring for ever. So, this recomended to catch and handle the Exception and also recomended to display non-technical Layman messages guiding the end user the exception is raised. don't display underlaying technology generated technical messages on browser coincidence when exception is raised. It is recomended to display them as non-technical error messages with proper exception handling.

Exception handling on Struts Action class can be done in two ways

① using programatic approach (use try and catch blocks in the execute(-,-,-,-) of Struts Action class)

② using Declarative approach (use <exception> tag of Struts Config file)

ⓐ global Exception handling (common for multiple Struts action classes)

ⓑ Local Exception handling (specific to each Struts Action class)

Procedure to enable programatic Exception handling on Struts application.

Step-I:- Keep 1st Struts appn ready.

Step-II:- write try & catch block as shown below in the execute(-,-,-,-) of Action class. (RegisterAction.java)

```
public AF execute(-,-,-,-)
{
  try
  {
     ==
     ==
  }
  catch(Exception e)
  {
    return mapping.findForward("err");
  }
}
```

Step-III:- add one additional result page configuration for Struts Action class in Struts Config file.

Struts-Config.xml

```
<Struts-Config>
  <form-beans>
  =
  </form-beans>
  <action-mappings>
  =
  <forward name="err" path="/Error.jsp" />
  =
  </action-mappings>
</Struts-config>
```

step-IV :- Develop the error page as shown below

Error.jsp

```
<font color=red> <center> !!! oh --- Some Internal problem </center> </font>
   <br> <br>
<a href= register.jsp > Try Again </a>
```

step-V :- Run the program in Reguler manner.

(*) procedure to perform Declarative Local Exception Handling.

step (I) :- keep 1st appn ready.

step (II) :- Develop the logics in execute(-,+,-,-) without try & catch blocks by just having throws Exception.

step (III) :- make sure that properties file is added to Struts appn

myfile.properties

```
my.exp.msg = <font color= red> <center> Internal problem </center> </font>
```

step (IV) :- Configure Declarative and Local Exception handling for RegisterAction class in struts-config.xml

```
<struts-config>
      :
<action-mappings >
  <action   path="/register"   type="app.RegisterAction"   name="rf" >
    <exception  type="java.lang.Exception"   key="my.exp.msg"   path="Error1.jsp"/>
      <forward  name="success"  path="/success.jsp" />
      <forward  name="failure"  path="/failure.jsp" />
  </action>
</action-mappings>
</struts-config>
```

(path="Error1.jsp" → errorpage name; key="my.exp.msg" → key in the properties file)

step (V) :- Develop the Error1.jsp as shown below.

Error1.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<html:errors />
```

step (VI) :- Run the program in Reguler manner.

Sample code in both Local & Global Exception handling in struts-config.xml

in struts-config.xml

```
<struts-config>
    <form-beans>
        =
        =
    </form-beans>

    <global-exceptions>
        <exception type="java.lang.Exception" path="Error.jsp" key="my.exp.
                                                    (global Exception Handling
                                                     Configuration).
    </global-exceptions>

    <action-mappings>
        <action path="/xyz" type="RegisterAction1" name="rf">
            <exception type="java.lang.Exception" path="Error1.jsp" key="my.
                                                    exp.msg1"/>
                                    (local Exception handling config for RegisterAction1 cla
        </action>
        <action path="/abc" type="RegisterAction2" name="rf1">
            <exception type="java.lang.Exception" path="Error2.jsp" key="my.exp.msg
                                    (local Exception handling config for RegisterAction2 class)
        </action>

    </action-mappings>

        =
</struts-config>
```

NOTE:- if both Declarative Local & global exception handling config's are done in same Action class with different setting (different error page, error msgs) then the settings done in Local Exception Handling Configurations will be applied.

→ if both programatic & Declarative exception handling config's are done on Struts Action class then the settings done in programatic Exception handling Configurations will be applied

<exception> tag of struts config file can handle only the exceptions raised in Action classes. To Handle the exceptions of FormBean class only by Catch block base programatic Exception Handling is possible

Example on FormBean class based Exception Handling

Step-I :- keep 1st appn ready.

Step-II :- write following code in Validate( ) of FormBean class

```
public ActionErrors Validate(ActionMapping mapping, HttpServletRequest req)
{
    try
    {
      int x = Integer.parseInt("ab123");   // Form Validation Logic.
      ActionErrors errs = new ActionErrors();
    } return errs;
    catch(Exception e)
    {
      ActionErrors err2 = new ActionErrors();
      err2.add("exp", new ActionMessage("my.exp.msg"));
      return err2;
    }
}
```

Creating a new validation error when Exception is raised

Step-III :- make sure that my.exp.msg key is there in properties file having error message

Same as step-III of previous discussion.

Step-IV :- Develop one error jsp program.

Error2.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
  from Error2.jsp
<html:errors />
```

Step-V :- configure the above error page as input page of Action class.

struts-config.xml

```
<action path="/register" type="app.RegisterAction" name="rf" input="/Error2.jsp" />
</action>
```

Step-VI :- Run the prg in Regular manner.

Exception Handling and Error page configuration in jsp is possible in two ways.

① Declarative Local Exception Handling (Specific to each jsp page) (use errorPage, isErrorPage attributes of <%@page %>)

② Declarative Global Exception Handling (Common for multiple jsp prgs of webappn) (use <error-page> of web.xml file)

(*) Procedure to perform Declarative Local Exception Handling in jsp.

Step-I :- keep 1st struts appn ready

Step-II :- write following code in any Jsp prg.

```
<%@ page errorPage="err1.jsp" %>
<% Class.forName("abc"); %>
<html>
 <body>
  <center> Login Success true </center>
 </body>
</html>
```

Step-III :- add err1.jsp

```
<%@ page isErrorPage="true" %>
 <i> from err1.jsp </i> <br>
 <b> internal problem </b>
```

~~Step-IV) Run the program in Eclipse server.~~

Step-IV :

(*) For Global Declarative Exception Handling write following entries in web.xml file.

in web.xml

```
<error-page>
 <exception-type> java.lang.Exception </exception-type>
 <location> /error2.jsp </location>
</error-page>
</web-app>
```

(for any exception raised in any Jsp program the control goes to error2.jsp)

error2.jsp :- // same as the above err1.jsp.

Step-V :- Run the appn

Note :- in both Declarative local, Declarative global Exception handling configurations are done in Struts Configuration file the settings done in Local Exception Handling Configurations will be applied.

08/01/2012

Q:- What is double posting problem?

Ans:- Clicking on Form Submit button continuously for multiple times (or) clicking on refresh button on the result page generated after submitting request creates "double posting problem".

The following are the Side effects related to Double posting problem.

*) Storing same User details for multiple times during User Registration.

*) Detecting amount for multiple times while doing online payments etc...

To solve this double posting problem use "Tokens" concept. that means create one token in Users Session & send that token along with the request and also reset the token Once request processing is Completed. So that we can check weather token is there (or) not there in FormPage generated request. if there process the request & if not there generate the error page. (make all normal request coming to Action class having token & make Double posting related request coming to ActionClass without token).

The following 3 methods of org.apache.struts.action.Action class are given to work with tokens.

① saveToken(req) ⟶ creates new Token in user's session.

② resetToken(req) ⟶ remove Token from Users session

③ isTokenValid(req) ⟶ checks weather Token is there (gives true) or not there (gives false) in Current user's session.

NOTE:- a Token is a flag in the request indicating that the Current request is normal request. so, the request without token Can be Treated as Invalied request. / Double posting problem

(*) For example application on demonstrating Double posting problem and Solving the problem through token refer the Suplimentary handout given on 08/01/2012

© sends implicit request to struts appn having dispatch = savingToken as requestParam value.

```
>>>>>>>>>>>>>StrutsApp on Double Posting >>>>>>>>>>>>>>>>>>>
----------------------------index.jsp--------------------------
<jsp:forward page="beforeRegister.do">
        <jsp:param name="dispatch" value="savingToken"/>
 </jsp:forward>
 ----------------------web.xml-------------------------------
 <web-app>
      <servlet>
           <servlet-name>action</servlet-name>
           <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
           <init-param>
                <param-name>config</param-name>
                <param-value>/WEB-INF/struts-config.xml</param-value>
           </init-param>
           <load-on-startup>1</load-on-startup>
      </servlet>

      <servlet-mapping>
           <servlet-name>action</servlet-name>
           <url-pattern>*.do</url-pattern>
      </servlet-mapping>

      <welcome-file-list>
           <welcome-file>index.jsp</welcome-file>
      </welcome-file-list>
 </web-app>
 ---------------------------------struts-config.xml-------------------------
 <!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
        "http://struts.apache.org/dtds/struts-config_1_3.dtd">
 <struts-config>
      <form-beans>
           <form-bean name="rf" type="app.RegisterForm"/>
      </form-beans>

      <action-mappings>
           <action path="/register" type="app.RegisterAction" name="rf" parameter="dispatch">
                <forward name="success" path="/success.jsp"/>
                <forward name="failure" path="/error.jsp"/>
           </action>
           <action path="/beforeRegister" type="app.RegisterAction" parameter="dispatch">
                          <forward name="success" path="/register.jsp"/>
           </action>
      </action-mappings>
 </struts-config>
 ---------------------------------register.jsp-----------------------------
 <%@taglib uri="http://struts.apache.org/tags-html" prefix="html" %>

 <html:form action="register">
   UserName:<html:text property ="username"/></br>
    Password:<html:password property="password"/></br>
   <html:hidden  property="dispatch" value="process1"/>
  <html:submit value="CheckDetails"/>
 </html:form>
 ---------------------------------RegisterForm.java--------------------------------
 package app;
 import org.apache.struts.action.*;

 public class RegisterForm extends ActionForm
 {
```

ActionServlet traps and takes (D) the request

This generates Home page

```
61      private String username;
62        private String password;
63        //write getXxx() and setXxx(-) methods
64        public void setUsername(String uname)
65        {
66        username=uname;
67        }
68
69        public void setPassword(String pwd)
70        {
71        password=pwd;
72        }
73
74        public String getUsername()
75        {
76        return username;
77        }
78
79        public String getPassword()
80        {
81                  return password;
82        }
83  }
84  ------------------------RegisterAction.java-----------------------------
85  package app;
86  import org.apache.struts.actions.*;
87  import org.apache.struts.action.*;
88  import javax.servlet.http.*;
89  public class RegisterAction extends DispatchAction
90  {
91  public ActionForward process1(ActionMapping mapping,
92                                        ActionForm form,
93                                       HttpServletRequest req,
94                                       HttpServletResponse res)throws Exception
95  {
96        System.out.println("process1(-,-,-,-) method");
97        System.out.println("Token valid"+isTokenValid(req));
98        if(isTokenValid(req))  → checks weather the current request is maintaining token or not
99        {
100        System.out.println("Processing B Logic");
101        Thread.sleep(10000);
102        RegisterForm fm=(RegisterForm)form;
103        if(fm.getUsername().equals("raja") && fm.getPassword().equals("hyd"))
104        {
105           req.setAttribute("msg","validCredentials");
106        }
107        else
108        {
109  when double Posting problem is not raised  req.setAttribute("msg","InvalidCredentials");
110        }
111        resetToken(req);
112  (O)   return mapping.findForward("success");
113  }
114        else
115  (O)   return mapping.findForward("failure");
116     }//process1
117
118  public ActionForward savingToken(ActionMapping mapping,
119        ActionForm form,
120        HttpServletRequest req,
```

(N)

when double Posting problem is raised

(118)-(127) → (F)

```
121         HttpServletResponse res)throws Exception
122         {
123        System.out.println("savingToken(-,-,-,-) method");
124         //register token for client .
125         saveToken(req);
126         return mapping.findForward("success");
127         }//savingToken                                    (G)
128   }//class
129   ----------success.jsp----------
130   <b><h1><center>ResultPage</center></h1><br>
131    Result is : <%=request.getAttribute("msg") %>
132
133   <p> <a href="beforeRegister.do?dispatch=savingToken">FormPage</a></p>
134   ----------error.jsp----------
135
136   <h1><center>Double Posting Not Allowed</center>
137   <p>
138     <a href="beforeRegister.do?dispatch=savingToken">Form Page</a>
139    </p>
140
```

(A) (B)

(B) (b)

TokenApp
- index.jsp
- error.jsp
- register.jsp
- Success.jsp
- WEB-INF
  - classes
    - *.java(2)
    - app
      - *.class(2)
  - lib
    - *.jar(10)
  - web.xml
  - struts-config.xml

jar files in CLASSPATH :-
servlet-api.jar, struts-core-1.3.8.

jar files in WEB-INF\lib folder
10 → struts jar files.

Request url : http://localhost:2020/TokenApp
(A)

⁂ Struts and EJB integration ⁂

Middlecware Services are additional, optional configigurable services to make our applications more perfect and accurate.

Eg:- Transaction mgmt, security, Login, JDBC connection pooling and etc.

→ middlecware services are not minimum logics of appn development. They are additional logics to configure on our appns.

→ while developing Struts appns keeping Business Logic and persistence logic in StrutsAction class is having the following limitations.

① logics become specific to one web appn.

② logics can be accessed only from those clients who can generate httprequest.

③ Adding middlecware services support on the logics becomes quite complex.

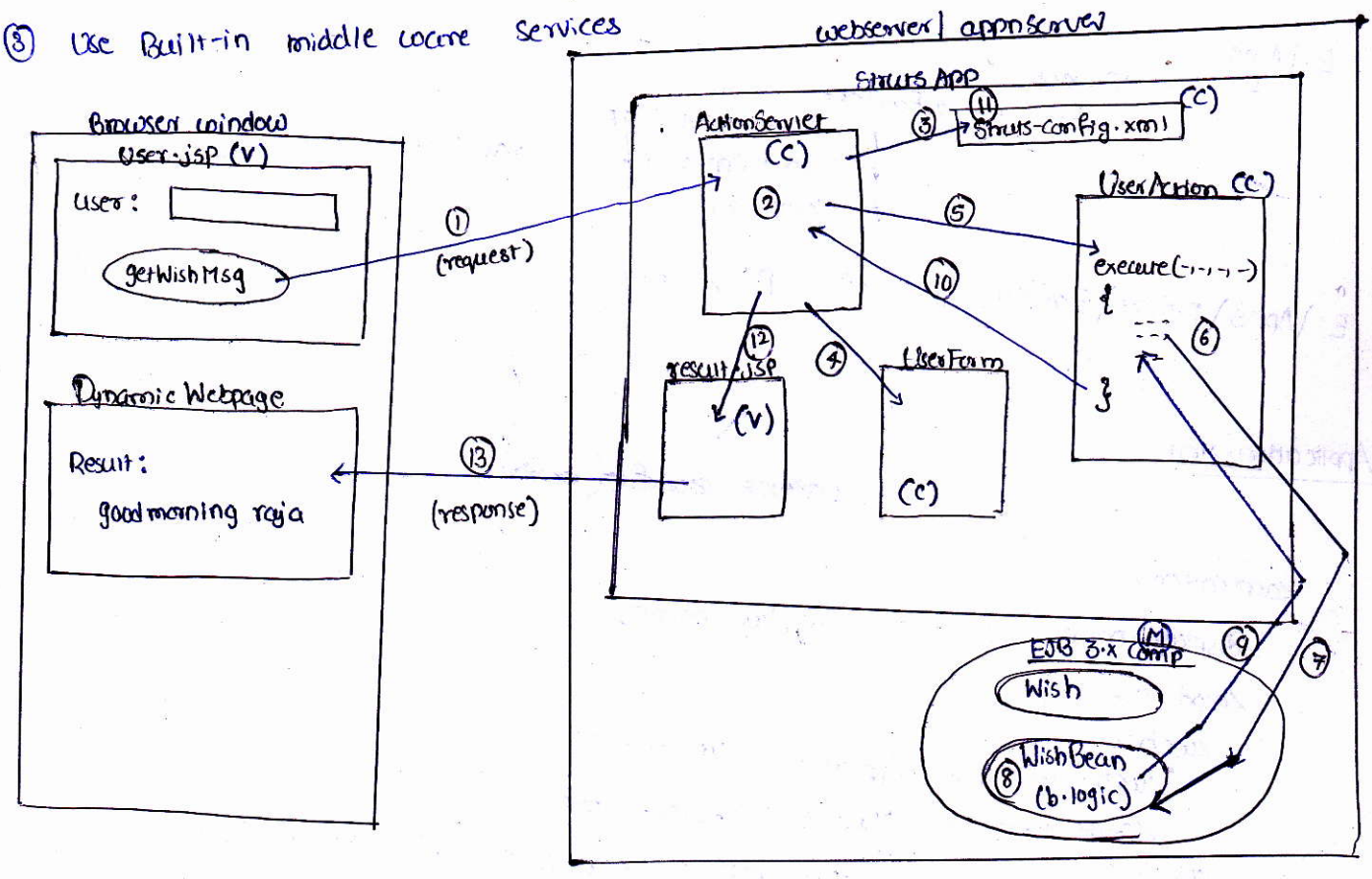
→ To solve these problems use EJB components, (or) spring with Hibernate appns to develop model layer Business logics, persistence logics

→ Using EJB components we can get following benfits.

① allows to develop reusable logics for multiple appns.

② allows both local, remote clients and http, non-http clients to access the logics.

③ Use Built-in middle cocre services



09/01/2013

→ In Struts to EJB communication the Struts Action class contain logic to interact with EJB component.

If Struts Action class contains BusinessLogics and persistence Logics directly then it is called Model Layer resource. If same Actionclass contains logic to interact with other Model Layer resources then it is called Controller Layer resource.

(*) For example appn on the above diagram based Struts & EJB integration refer application (13) of the page no's (100) to (102)

(*) procedure to deplogee and execute application (13) of the Struts Booklet.

Approach 1:- deplogee war & jar files seperately in any domain server of GlassFish.

Copy comp.jar, StrutApp.war files to <GlassFish_home>/appserver/domains/domain1/autodeplogee folder.

Start domain1 server (Start → programs → Sunmicrosystems → appnServer → Startdefault server)

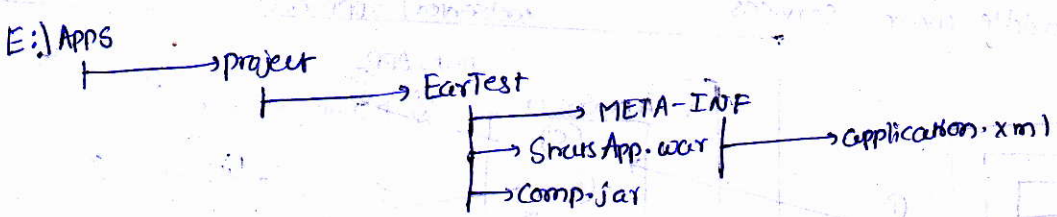
http://localhost:2020/StrutsApp/user.jsp

Approach 2:- ear file = jarFile + jarFile + ...

ear file = warFile + warfile + ...

ear file = jarfile + warfile + jarfile + warfile + ...

For Earfile application.xml is Deplogeement Descripter file which cfgs of warfile, jarfiles..



E:\Apps\project\EarTest> jar cf Project.ear

Application.xml

~~Deplogee ear file(note~~

```
<application>
  <display-name> EARFile </display-name>
  <module>
    <web>
      <web-uri> StrutsApp.war </web-uri>
      <context-root> /MyWeb </context-root>
    </web>
```

context path of the webappn to be used while accessing the webappn.

```
</module>
<module>
    <ejb> comp.jar </ejb>
</module>
</application>
```

Deployee ear file (copy project.ear file to <GlassFish_home>/appserver/domain/domain1/autodeployee folder)

request url:-

http://localhost:8080/MyWeb/loishact.do

└ Context path of the webapp given in application.xml file

→ Struts 1.x ←

Struts 2.x

- → vendor: Apache Foundation
- → version: 2.1.x (compatable with jdk1.5+)
- → open source web f|w
- → To download s/w: as zip file from www.apache.org
  - zip file: Struts 2.0.11-all.zip
  - Struts 2.3.4-all.zip
- →

Struts 2.x = Struts 1.x + webwork

10/01/2013

limitations of Struts 1.x :-

→ Struts API is having more Abstract classes. so, it is not a good API.

→ The java resources of Struts appn Struts API dependent. so, Unit testing of these resources (Programmers Testing) is complex.

→ working with Validator plug in to perform form validations is quite complex.

→ working with checkboxes & listboxes needs some additional programing.

→ There is no Annotation Support.

→ View layer allows only HTML, JSP technologies to develop presentation logics

→ Debugging of Struts application quite complex.

→ Action classes are not Thread safe by default. so, we need to explicitly ego for Synchronization.

→ and etc.....

Annotations are Java statements which are alternate for the xml files based resource configurations and metadata operations.

→ XML files give good flexibility of modification but bad performance (bcz xml parsers are heavy weight slds

→ Annotations give good performance and Bad flexibility of modification

→ Annotations are introduced from Jdk 1.5 for programing. so, The technologies that are designed based on Jdk 1.5 + are giving support for Annotations

These technologies are Struts 2.x, EJB 3.x, Spring 2.5+, Hibernate 3.3+ and etc...

Syntax of Annotations :-

@<annotation-name>(param1 = value1, param2 = value2, ...)

(param1 = value1 → like xml attribute)

(<annotation-name> → It is like xml tag name)

→ Struts 2.x supplies only one tag library having 3 types of tags.

(1) Generic tags (2) User Interface tags (3) Ajax tags

→ Struts 2 is designed based on Servlet API 2.4, JSP API 2.0, Jdk 1.5.

→ <Struts2-home>/lib/struts2-core-2.0.11(version).jar, xwork-version.jar files represents Struts 2 API's.

(x) For Features of Struts 2.x refer Page no: (107) of the Booklet. :-

→ an ordinary Java class without any Technology specific API dependency is called "POJO" class. an ordinary Interface without any Technology specific API dependency is called "POJI".

→ Struts 2 Action class is POJO class because it can be developed without extending, implementing class, interfaces of Struts 2 API.

Q:- What is the difference blw Struts 1.x and Struts 2.x environment ?

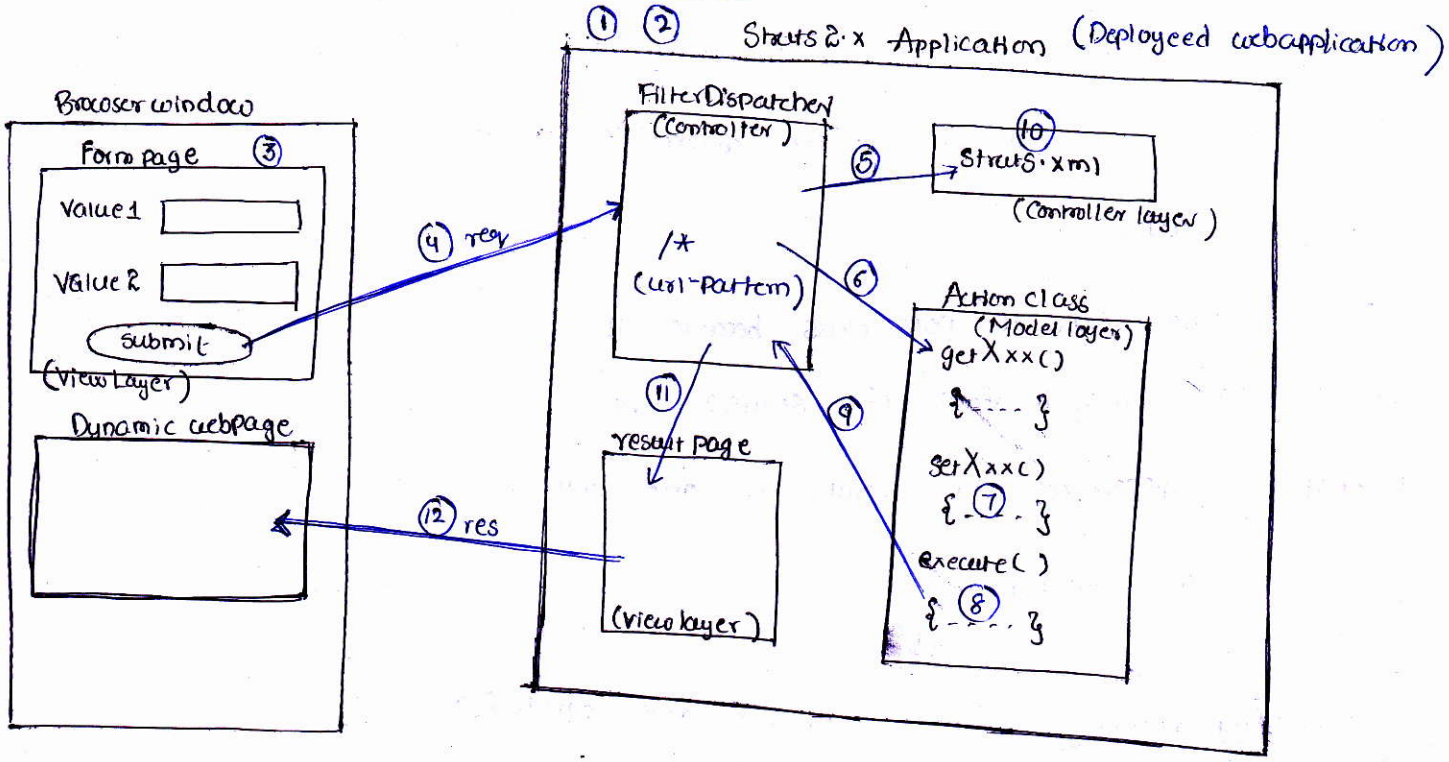
refer page no's 108 & 109.

11/01/2013

Terminology matching between Struts 1.x and Struts 2.x

	Struts 1.x	Struts 2.x
Controller	Action Servlet (org.apache.struts.action pkg)	FilterDispatcher (org.apache.struts2.dispatcher
	FormBean	.....
	ActionForwards cfg	result cfg
	Struts Action class (non-pojo class)	Struts Action class (can be a Pojo class)
	Action Mapping	Packages
Customizing Controller	user-defined Request Processor	user-defined Interceptors
Struts cfg file	<Any file name>.xml (any location)	Struts.xml (WEB-INF/classes)

Flow of execution in Struts 2.x application :-



→ If Action class contains Business Logic directly then it comes under "Model layer resources"

→ If Action class contains Logic to interact with other model layer resources like EJB component, Spring with hibernate appn and ~~extra~~ etc then it comes under "Controller Layer resource"

→ Struts 2.x Action class object will not be created or located when the Form page is launched on the Browser window.

→ Servlet Filters are having default behaviour of getting instantiated either during Server startup or during the deployement of webapplication automatically.

So, there is no need of enabling <load-on-startup> on Servlet Filter.

W.r.to the diagram

① Programmer deployees Struts 2.x appn in the server

② Servlet container creates object of FilterDispatcher either during Server startup (or) during the deployement of the webappn. In this process FilterDispatcher reads and verifies Struts.xml file implicitly.

③ Enduser launches the Form page of Struts 2.x appn on Browser window.

④ End user submits the request from Form page & as controller FilterDispatcher traps & takes the request

⑤ FilterDispatcher uses Struts.xml file entries to decide the Action class that has to be used to process the request.

⑥ FilterDispatcher creates Struts Action class object.

⑦ FilterDispatcher calls setXxx( ) of Action class to copy Form data to Action class property.

⑧ FilterDispatcher calls the execute( ) of Action class where Business Logic is available to process the request.

⑨ execute( ) return String value to FilterDispatcher.

⑩ FilterDispatcher uses results Configuration to decide the result page of Action class.

⑪ FilterDispatcher passes the control to Result page

⑫ presentation logic of Result page format the results and sends Formatted results to Browser window as Dynamic webpage content.

NOTE:- while Completing ⑤ to ⑪ operations FilterDispatcher internally uses interceptors.

we configure FilterDispatcher in web.xml file having /* url-pattern to make FilterDispatcher as Controller and to allow FilterDispatcher trapping all the request from coming to Struts 2.x appn.

web.xml (Struts 2.x appn)

```
<web-app>
  <filter>
    <filter-name> Struts2 </f-n>
    <filter-class> org.apache.struts2.dispatcher.FilterDispatcher </f-c>
  </filter>
  <filter-mapping>
    <filter-name> Struts2 </f-n>
    <url-pattern> /* </u-p>
  </filter-mapping>
</web-app>
```

→ Struts-default.xml file given by Struts 2.x contains some Built-in result types, interceptors and Interceptor stacks ~~Interceptor tags~~.

→ This file contains 11 result types like chain, dispatcher, stream... and etc which are useful to generate result pages to Browser window.

This file contains 30 no of Interceptors lik alias, Validation, prepare, scope, token and etc

These interceptors are quite useful for FilterDispatcher to decide the Flow of execution request coming to related to Action class.

→ Struts-default.xml file also gives 10 interceptor stacks where related predefined interceptors are Combined into Single unit.

These Interceptors tags are like BasicStack, defaultStack, executeAndWaitStack, and etc...

Interceptors are Configurable on per Action class basis.

→ "defaultStack" interceptor Stack contains "16" interceptors which can Controle the Basic flow of execution of Action class.

→ If no Interceptor is explicitly Configured every Action class of Struts 2.x will be Configured with the interceptors of "defaultStack" by default.

→ Struts 2.x package is a logical unit where set of Action classes, Result Pages, Interceptors can be Configured.

→ Struts 2.x Package terminology is no way related to the Java pkg terminology.

→ to make Action classes of our struts.xml file working with interceptors of default Stack, Other interceptors and to make these Action classes also using Result types (Built-in Result types) we need to perform the following two operations.

① include Struts-default.xml file in struts.xml file.

② create new packages in struts.xml file by extending from struts-default package. as shown below.

in struts.xml

```
<!DOCTYPE ----------
          ----------/Struts-2.0.dtd">

<Struts>
  <include file="struts-default.xml" />

  <package name="pack1" extends="struts-default">
    <action name="display" class="P1.DisplayAction">
      <result name="success">/success.jsp</result>
      <result name="failure">/failure.jsp</result>
    </action>
  </package>

</Struts>
```

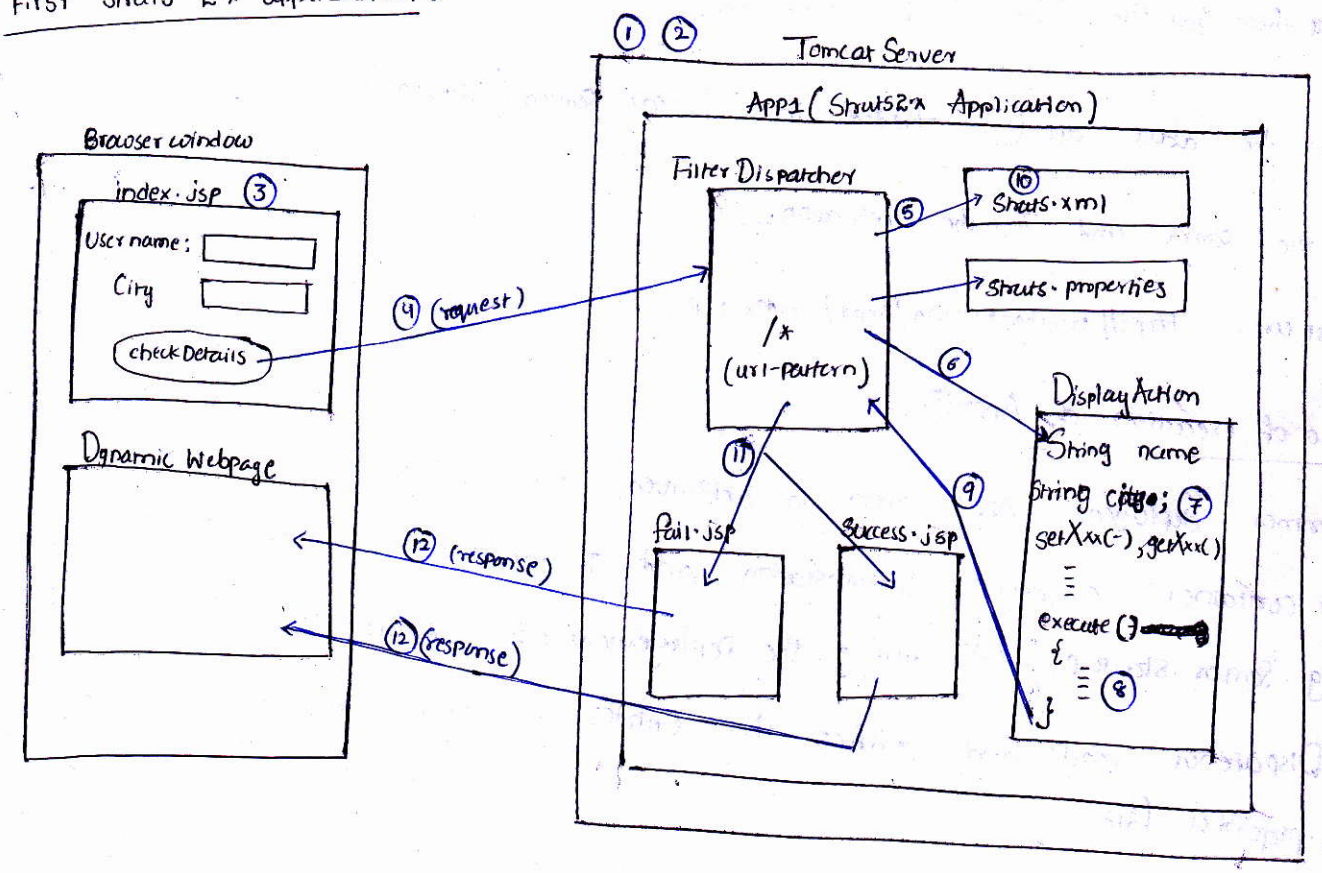
- `pack1` → our package name
- `struts-default` → package name in struts-default.xml file
- `display` → logical name
- result pages configuration
- action class configuration

12/01/2013

In Struts 2.x apps there can be fixed struts.properties file to specify various parameters like user defined property file names, cache flag, defaultLocale and etc.....

The location of struts.properties file is WEB-INF\classes folder.

First Struts 2.x application :-



→ Struts 2.x s/w supplies only one built-in Jsp tag library called Struts tags. Its tld file is "struts-tags.tld" (available in struts2-core-2.0.11.jar)

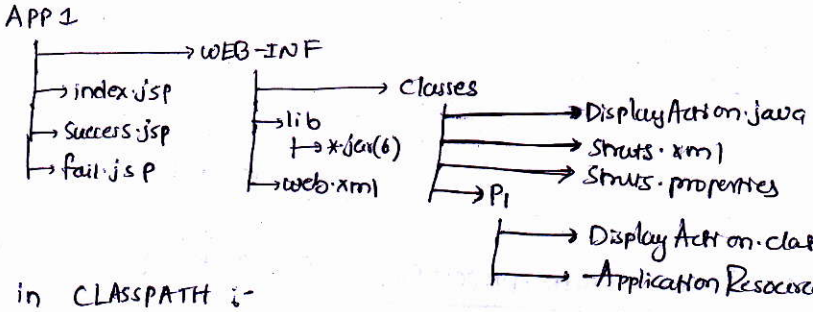
→ The taglib uri is "/struts-tags"

(*) For the above diagram based appn Source Code refer App 1 of page no's (110) to (112)

procedure to develop, deploy and execute Struts 2.x application (App 1 of page no's 110 to 112)

① extract <Struts2.x_home>\apps\ struts2-blank-2.0.11.war file to a temporary folder.

② use the above extracted content as reference content and create the deployement directory structure of our Struts application.



jar files in CLASSPATH :-

no jar files are required. (Because DisplayAction class is POJO class).

Jar files in WEB-INF/lib folder :-

commons-logging-1.1.jar, freemarker-2.3.8.jar, ognl-2.6.11.jar, struts2-core-2.0.6.jar, xwork-2.0.1.jar

gather all these jar files from step-I extraction.

③ Deploye the above appn① folder in Tomcat Server (Tomcat 5.5+).

④ start the server and Test the web appn.

Request url: http://localhost:2020/App1/index.jsp

The flow of execution of Appn① :-

(A) programmer deployees App1 appn in webserver (or) Application Server

(B) Servlet container completes instantiation and Initialization of FilterDispatcher either during Server startup (or) during the Deployement of webappn

(C) FilterDispatcher reads and verifies the Entries of struts.xml file and the struts.property file.

(D) End user gives request to App① appn

http://localhost:2020/App1/Index.jsp (④ - ㉓)

(E) End user submits the request from Form page having the data (refer ⑱)

(F) Based on the Action url placed ~~based~~ in <S:form> tag the following request url will be generated.

(refer: ⑮)

http://localhost:2020/App1/display.action

(G) The FilterDispatcher traps and takes the request (59 - 67)

(H) based on display url of request url FilterDispatcher locates DisplayAction class configuration in struts.xml file whose logical name is "display" (refer 45 - 48)

(I) FilterDispatcher creates DisplayAction class object using 0-param constructor support (on one per request basis).

(J) FilterDispatcher called setXxx(-) on DisplayAction class object and copies the received form data to Action class property. (refer 87 - 89 & 96 - 99)

(K) FilterDispatcher calls execute() on the Object of Action class (refer 101 to 114)

(L) execute() returns one result configuration result logical name (refer 107 (or) 111)

(M) FilterDispatcher receives value given by execute() method and uses that value to decide the result page of Action class based on results configuration. (refer 46 or 47).

(N) FilterDispatcher passes the controle to one Result page. (refer 117 - 131 (success.jsp) 132 to 147 (failure.jsp)

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