**Struts Framework**

Struts mainly used for providing the support on View and Model side of MVC Architectural pattern

For View side it will provide following support

1. **Form Tag support**: If we use simple html tags form backup and validations are not supported while if we use Struts forms tag then we can have support for form validations and form backup support.
2. **I18N Support**: It provides support for internationalization ie supports for multiple languages.
3. **Logical Tag Support**: For iterating the values of arrays and collection types of the database into the form we can use struts logical tags. We can also iterate to format the table i.e. its color row wise. It also consists of conditional tags.
4. **Tiles Tag Support**: Using this we can use tiles tag support to provide for template support.

Controller

Model

View

Struts is used for these two layers.

For Controller layer we will have following types of support

1. **Multi Action Controller:**
2. **Multi Button Controller:** If we are applying I18N support to our application then we cant use MultiButton Controller as it works on button values and various database CRUD operations operation will be written inside the method name same as button value
3. **Multi** **Form Controller:** If we are using three forms then we can store the data of all three beans in a single bean object and can persist into the database.
4. **Exception Handler Support**: if our application throws any exception then it will call the required handling code. In case of struts application we do not have to compulsorily throw compile time exceptions like IOException, ServletException, ClassNotFoundException etc.
5. **Plug in support**: plug in class are given to integrate controller with model part ie integrating struts with hibernate or Spring.
6. **Pre Handling support**: If we want to execute something before our controller class runs then we can make use of pre handling classes just like filters like security, pre-loggings. It is possible in struts 1x only. Like if we have thousand forms and we want to provide the security for 1000 forms then we can provide for the same by using pre handling classes. In struts 2x we have **post handling support** also in the form of interceptors which is useful for form rendering for output. We also have AJAX tag support and OGNL(Object Graph Notation Language) support.
7. **I18N Controller** classes to change the browser language.

For form backup and validations support we need to use following Helper classes

1. **Form Beans** for Form Backup
2. **Validators** for Form Validations

Different types of JSP Model

1. Jsp model 1: Using Jsp as controller
2. Jsp model 2: UsingServlet as controller for example are Struts1x(ActionServlet), JSF(FacesServlet), SpringMVC(DispatcherServlet) etc.
3. Jsp model 3: Using Filter as controller for example Struts 2.x (FilterDispatcher and StrutsPrepareAndExecuteFilter)
4. Jsp model 4: Using Tags as controller

If we submit any form request the front controller that is ActionServlet in case of Struts 2 will provide the support for validations, form backup, exception handling etc and user specific operations we need to write the user specific controller classes.

ActionServlet (Front Controller)

StudController

Stud\_Reg

EmpController

Emp\_Reg

LoginController

Login Form

From backup support can be given by following classes

1. ActionForm
2. DynaActionForm
3. ValidatorForm
4. DynaValidatorForm
5. ValidatorActionForm
6. DynaValidatorActionForm

Controller class can be implemented using following classes

1. Action
2. DispatchAction
3. LookupDispatchAction
4. MappingDispatchAction
5. ElementDispatchAction

ActionServlet

Struts-config.xml

doGet(req, res)

doPost(req,res)

Login Form

Emp\_Reg

Stud\_Reg

Init(ServletConfig sc){

String xml=sc.getInitParameter();

SAX

Process(req,res) ModuleConfig

RequestProcessor rp= RequestProcessorUtil(mc) }

Rp.process(req, resp)

processFormBean(req, res) RequestProcessor

Process(req, res)

Boolean processpreprocess(req,res)

processLocale(req, res)

Bean

processValidation(req, res)

processController AE ae = bean. Validate();

(req, res) If (ae.icount==0){} else{

Rd.forward(req, res)}

ModuleConfig will store the xml files for each module if we have many modules in our application and will be provided to intantiate the helper class of ActionServlet name named as RequestProcessor which is a singleton object. For each module struts will create separate RequestProcessor object. On RequestProcess Object it will call process method which will delegate to the process method its task and will execute the processpreprocess(req, res) which further on returning true will call processLocale(req, resp) method which will read the browser language as follows

String lang= req.getHeader(“accept-language”);

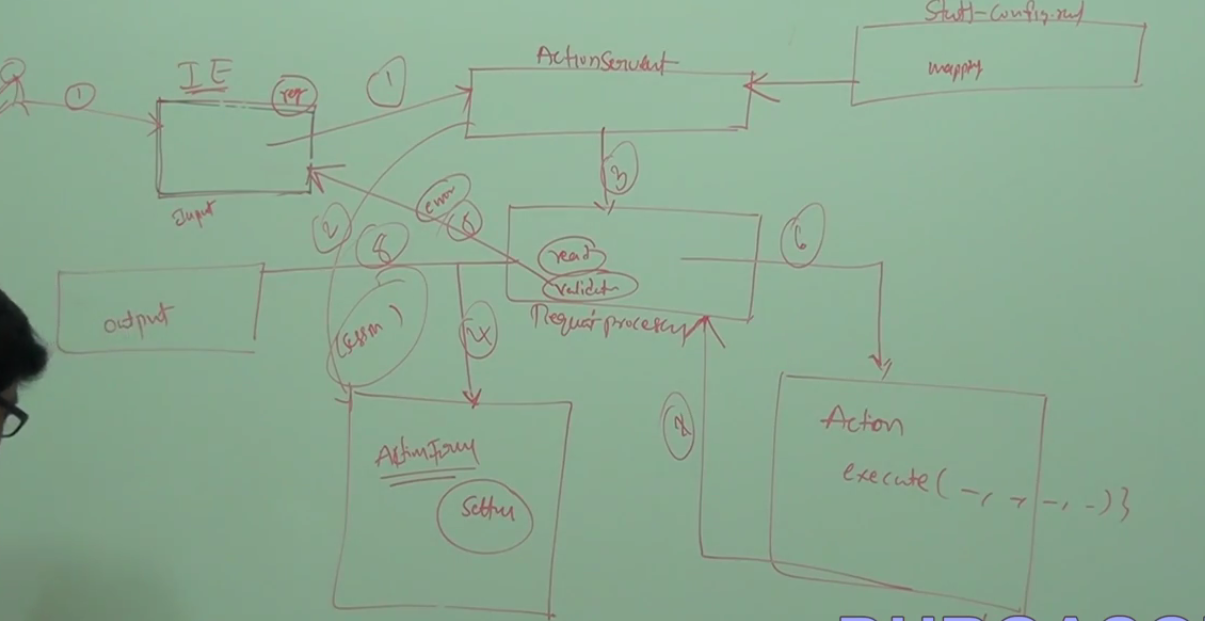
Locale l = new Locale(lang);

Session.setAttribute(“org.apache.struts.locale”, l);

The above code will be executed only for one time for entire application per session.

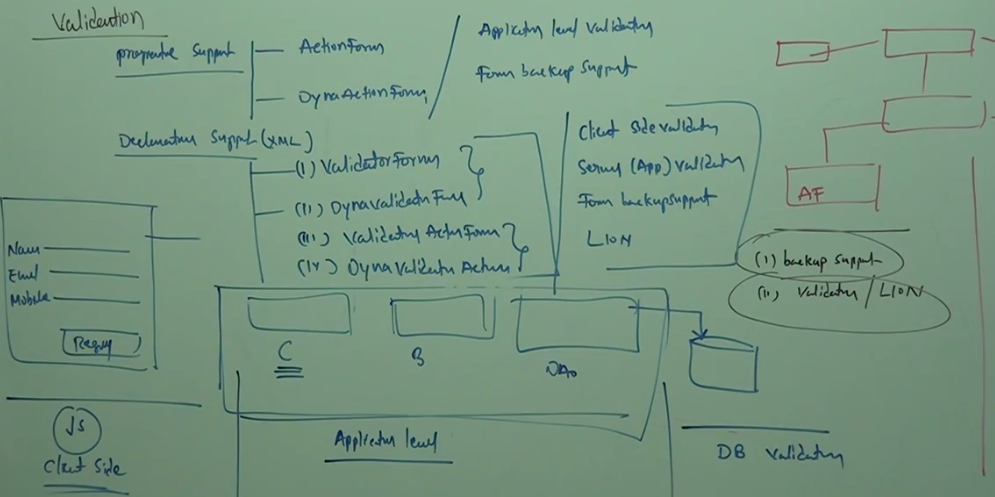
processFormBean method will store the data from form into Bean object and will provide the backup support.

RequestProcessor will process method will also call processValidate() method which will execute the validations on form and will also call processCOntroller which will call the execute() method of Controller class.



Architecture Of Struts

Various types of Validation that can be performed in strutss framework



If we set scope of bean object in action mapping tag as session then for each user it will create a single bean object but when if we set it to request scope then it will create the bean object for each request for example if a user hit the submit button for several times in order to enter every values in the form correctly then for each request it will create a separate bean object which is not good in programing point of view.

Validations are performed in following ways

1. Programmatic
2. **ActionForm** Validations and FormBackup Support
3. **DynaActionForm** Generally used for Form Backup Support
4. Declarative Approach (Single Form handlings)
5. **ValidatorForm** Client(JS will generated automatically)/Server Side Validation
6. **DynaValidatorForm** Form Backup Support and Localization Support

MultiformHandling

1. **ValidatorActionFrom**
2. **DynaValidatorActionForm**

In case of Declarative approach we need to pass the xml file for validations unlike in Programmatic approach where we need to use Java Validations to validate the form data. We need to have 2 xml files validation-rules.xml and validator.xml and to read these xml files we need to configure these in plug in tag of struts-config.xml file. At the time of deployment init() method inside ActionServlet will read struts-config.xml file data. By using configure our validator plug-in class inside plug-in tag of struts-config.xml so that these xml files are parsed for validations.

If our Bean class extends ActionForm and DynaActionFrom then validate method will be called by RequestProcessor object. But if our Bean class is instance of ValidatorForm, DynaValidatorForm, ValidatorActionForm and DynaValidatorActionForm then it will not call any validations here but ValidatorPlug-in Class will read the validations from Xml files and will place that data in servlet context scope. RequestProcessor will now read the validation from context scope and will apply the same on Data stored inside the bean. If any validation errors are there then It will be returned to the index page otherwise then control will be send to the Action(Controller) class. One more advantafe with these xml file is that it will generate the JavaScript Validations for client side validations.

These validations done by validation.xml and validator-rules.xml are doing server side validations which can cause huge traffic if website is being accessed by a lot of users and can affect the performance of the application which is a not good programming methodology. Its better to generate the Javascript code for validations purposes so that validartions can be done at client side only which reduce a huge amount of traffic and hence will increase the efficiency of web application. We need to add following tag to generate the javaScript code for validations in our jsp page

**<html:javascript formName=”beanReference”/>.**

A JavaScript with function name as validate<beanreference> will be generated which we need to call onsubmit value of form ie when form is submitted

<html:form action=*"reg"* method=*"post"* onsubmit=”*validateRF*”>

**What is difference between ActionForm and DynaActionForm?**

An ActionForm represents an HTML form that the user interacts with over one or more pages.

You will provide properties to hold the state of the form with getters and setters to access them.

Whereas, using DynaActionForm there is no need of providing properties to hold the state.

Instead these properties and their type are declared in the struts-config.xml.

The DynaActionForm bloats up the Struts config file with the xml based definition. This gets annoying as the Struts Config file grow larger.

The DynaActionForm is not strongly typed as the ActionForm. This means there is no compile time checking for the form fields.

Detecting them at runtime is painful and makes you go through redeployment.

ActionForm can be cleanly organized in packages as against the flat organization in the Struts Config file.

ActionForm were designed to act as a Firewall between HTTP and the Action classes, i.e. isolate and encapsulate the HTTP request parameters from direct use in Actions.

With DynaActionForm, the property access is no different than using request.get Parameter( .. ).

DynaActionForm construction at runtime requires a lot of Java Reflection (Introspection) machinery that can be avoided.

Init()

Struts-config.xml

ActionServlet

input

Process()

RequestProcessor

creates errors

Corresponding Controller

validate()

Bean Object

AF, DAF, VF, DVF, VAF, DVAF response

Action

Mapped output View

DispatchAction

LookupDispatchAction

MappingDispatchAction

EventDispatchAction

Internationalization (I18N) support: Strust will provide the I18N support for upto 200 languages

Localization Support(L10N) support will also provide the language support.

If we write our application in struts language and we have made our web application in English language then our application will store the English locale at session scope by taking the default language of our browser. In order to provide the support for other language we need to override the method which stores the system language at first run only so that on user action that method is overridden and a new language is selected each time. If we are providing support for 3 languages then we have to write 3 properties file with appending the properties file name with appropriate extension as per ISO standards for example for English en, hindi hi, Telugu te etc.

Internally Struts will run following code that is inside the processLocale() method of RequestProcessor class

String lang= req.getHandler(“accept-language”);

Locale l= new Locale(Lang);

Session.setAttribute(“org.apache.struts.action.Locale”, l)

**MultiButton Handling using DisapatchAction Controller Class**

Aadhar Reg Form

ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Email:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Address:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Select All

Select

Delete

Update

Save

If we perform CRUD operation by suning the values of Form then button name is taken as method name and DispatchMaehod(am, af, req, resp, submit) of DispatchAction Controller class will take the submit button value and by using Java Reflection API it will try to create a method name with same submit value and into that method it will try to pass all these parameters say it save(am, af, req, resp) having Parameters as ActionForward, ActionMapping, HttpServletRequest and HttpServletResponse then it will call the same method and will perform the operations written inside the save method. Here we should not override the execute() and Dispatch() Method. In case of using DispatchAction out method name should be same as that of Button Names. But if we are giving support for Internationalization support then Its not feasible. Because the dispatchMethod() of DispatchAction Controller class will not find the nethod with same name if language is changed and will throw NoSuchMethodException at runtime.

DispatchAction Controller class will not support I18N Support while LookupDispatchAction and EventDispatchAction will support I18N Support.

If we use LookupDispatchAction Class to implement our Controller class then method execute() of LookupDispatchAction will first check the language of submit button and after checking the language of button it will take the corresponding key value and getKeyMethodMap() method will find the key for that value and which returns a map object and in that map object in that map object we can have K,V pairs as Key and the method name that must be passed to dispatchMethod() so that method with same name is created at runtime using Java Reflection API and method is executed

Protected Map getKeyMehodMap(); means getKey and Map Method.

It will find the value of Button after that will find the key for the value.

Here we can use method name according to our wish there is not a restriction that we need to use same name as that of Key.

Using **EventDispatchAction** Controller Class

EventDispactchAction Class uses the Button Id of buttons unlike button value in DispatchAction Controller Class. In case of DispatchAction class in struts-config.xml file we need to provide in parameter of action tag as button value. But in case of **EventDispatchAction** class we need to maintain separate button id for each button and in parameter of action tag we need to write all these button id’s separating each of them with commas so that at runtime it will perform substring operation to get the button Ids and hence a method name with same button id will delegated to perform the action.

**MappingDispatchAction** Controller class will help to map the various form action to be mapped to a single controller class and the method name in controller class can be mapped to action name of that form in struts-config.xml so that specific method can be executed on form submit button click. So in parameter of action mapping tag we need to write the method name which must same as that of method inside controller class.

Along with User controllers struts given Built-In-Controller class(Prdefined Action Classes)

1. **ForwardAction Class**: Forwarding request from one jsp to other jsp and one controller to another controller.
2. **IncludeAction Class:** One form content it will include into other form.
3. **LocaleAction Class**: It will change our Browser Langugae
4. **DownloadAction Class**: Page Downloads, document Downloads
5. **ExceptionHandler Class**
6. **SwitchAction Class**: To switch from one module to another module we use SwitchAction Classes

FA

SwitchAction

FA

Form1

Form2

Employee

Admin

To switch our request from one module to other module then we use this Built-In Actions.

1. ForwardAction Class

We should not use direct link our jsp page to which we want forward our request otherwise the person browing website can see the url of the page in our <a href> tag and modify the page if it contains the business logic. So we should hide complete page urls.

So instead of using <a href=”/send.jsp”>send</a> we should use

<a href=”./send”>send</a> or <html:link action=”send”>Click</html:link> if we are using struts tag

1. Include Action Class; If we want the content of previous file in the second file then we use this Acion Class
2. SwitchAction Class; When we want to forward request from one page module to another page in other module then we use this Action Class and we can also use this class when we want content of one page in one module in a page of other module.

EMS(Employee Management System Application File Structure)

Admin

Employee

Users

WEB-INF

Classes Admin

Employee

Users

lib

web.xml

struts-config.xml : for Default Module

struts-admin.xml

struts-emp.xml

struts-users.xml

Appname/admin/reg.do

Login Form

If we run application on server then we need to have provide the module name then action name so that Registeration can happen successfully for each module. In our case we have three module employee, admin, users and if any type of users register then we need to move to respective module and need to hit Register button and according to the action of the form struts will execute the backend code as configured in struts config file.

**javax.servlet.jsp.JspException: Cannot retrieve mapping for action: "/adminupdate"**

We will get this error if our page contains the action that is not configured if switch from one module to another module then we have to edit the action mapping so that form of one module is forwards request to the form of another module. And we need to use SwitchAction.

For Modularization we need to send two attributes one is **prefix** name as name of the module and other is other is **page** represents the page actions.

<action-mappings>

<action path=*"/login"* name=*"LF"* input=*"/login.jsp"* type=*"com.ds.ems.main.controller.LoginAction"*>

<forward name=*"samepage"* path=*"/login.jsp"*/>

<forward name=*"adminsuccess"* path=*"/adminsuccess.do?prefix=/admin&amp;page=/adminforward.do"*/>

<forward name=*"employeesuccess"* path=*"/employee/employeesuccess.jsp"*/>

<forward name=*"usersuccess"* path=*"/user/usersuccess.jsp"*/>

</action>

<action path=*"/adminsuccess"* type=*"org.apache.struts.actions.SwitchAction"*>

</action>

</action-mappings>

Above struts config means whenever struts get “adminsuccess” forward then it forwards to adminsuccess.do action which is forwarded by SwitchAction to action **/adminforward.do** in **admin module** and this action we need to configure in **struts-admin.xml** as follows

<action-mappings>

<action path=*"/adminforward"* forward=*"/adminsuccess.jsp"*/>

</action-mappings>

<init-param>

<param-name>config/admin</param-name>

<param-value>/WEB-INF/struts-admin.xml</param-value>

</init-param>

If we want to change password form from login page only then we have to do the following things in login.jsp

<html:link action=*"adminforward"* module=*"/admin"*>Click to Redirect Admin</html:link>

DownloadAction Class usage

Struts.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>

<action-mappings>

<action path=*"/file"* type=*"controller.FileDownloadController"*/>

</action-mappings>

</struts-config>

File DownloadController

**package** controller;

**import** java.io.File;

**import** javax.activation.MimeType;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.apache.struts.action.ActionForm;

**import** org.apache.struts.action.ActionMapping;

**import** org.apache.struts.actions.DownloadAction;

**public** **class** FileDownloadController **extends** DownloadAction {

@Override

**protected** StreamInfo getStreamInfo(ActionMapping am, ActionForm af, HttpServletRequest req,

HttpServletResponse resp) **throws** Exception {

String filename=req.getParameter("filename");

StreamInfo info= **null**;

**if** (filename.contains("strutsbook")) {

File book= **new** File("D:/New folder/Wiley.Java.In.60.Minutes.A.Day.May.2003.ISBN.0471423149.pdf");

info= **new** FileStreamInfo("application/pdf", book);

resp.setContentType("application/pdf");

resp.setHeader("Content-Disposition", "attachment;filename=javabook.pdf"); //when application downloads the file then file name is given javabook.pdf as name

}

**if** (filename.contains("pkmovie")) {

File book= **new** File("D:/New folder/Wildlife.wmv");

info= **new** FileStreamInfo("application/wmv", book);

resp.setContentType("application/wmv");

resp.setHeader("Content-Disposition", "attachment;filename=pkmovie.wmv");

}

**if** (filename.contains("myimage")) {

File book= **new** File("D:/New folder/Paytm\_statement-May-2016.jpg");

info= **new** FileStreamInfo("application/jpeg", book);

resp.setContentType("application/jpeg");

resp.setHeader("Content-Disposition", "attachment;filename=paytm.jpg");

}

**return** info;

}

}

Index.jsp

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

<pre>

<html:link action=*"file.do?filename=strutsbook"*>Download Struts Book</html:link>

<html:link action=*"file.do?filename=pkmovie"*>Download PK Movie</html:link>

<html:link action=*"file.do?filename=myimages"*>My Image</html:link>

</pre>

Usage of LocaleAction Controller Classes

For changing our browser language we don’t need to write the controller class of our own. We can use LocaleAction class to change the browser language. Just we need to maintain the bean Class with language, Country and page. In index page along with action we need to pass value for language, country and page on which we want Localization to be applied.

Struts.xml

<struts-config>

<form-beans>

<form-bean name=*"LF"* type=*"org.apache.struts.action.DynaActionForm"*>

<form-property name=*"language"* type=*"java.lang.String"*/>

<form-property name=*"country"* type=*"java.lang.String"*/>

<form-property name=*"page"* type=*"java.lang.String"*/>

</form-bean>

<form-bean name=*"AF"* type=*"beans.AadharForm"*></form-bean>

</form-beans>

<action-mappings>

<action path=*"/reg"* name=*"AF"*/>

<action path=*"/locale"* name=*"LF"* type=*"org.apache.struts.actions.LocaleAction"*>

<forward name=*"success"* path=*"/index.jsp"*/>

</action>

</action-mappings>

<message-resources parameter=*"resources/AadharForm"*/>

</struts-config>

Index.jsp

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

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

<div style="border-color:*maroon*; color: *red*; font-size: *50*" >

<center><bean:message key=*"title"*/></center>

</div>

<html:form action=*"reg"*>

<pre>

<bean:message key=*"id"*/> <html:text property=*"id"*/>

<bean:message key=*"name"*/> <html:text property=*"name"*/>

<bean:message key=*"email"*/> <html:text property=*"email"*/>

<bean:message key=*"address"*/> <html:textarea property=*"address"*/>

<html:submit><bean:message key=*"submit"*/></html:submit>

<a href=*"./locale.do?language=en&country=IN&page=/index.jsp"*>English</a>

<a href=*"./locale.do?language=hi&country=IN&page=/index.jsp"*>Hindi</a>

</pre>

</html:form>

ExceptionHandlerAction

**Plug ins**

Plug-in support: Struts provided this support in order to integrate with other frameworks: We used apache commons validator in order to validate our form in one of our previous examples. We user validator-rule.xml and validation.xml. validation.xml file contained validations of each attribute on our form and validator-rules.xml file contains 14 rules by using which validations in validation.xml will be done.

We have following methods inside our Plug-In interface

1. Public void init(ActionServlet as , ModuleConfig mc);
2. Public void destroy();

When these Plug-in will come into action. First ActionServlet class will be instantiated by struts Container on it init method will run which will read struts-config.xml and if it finds plug-in configuration inside the struts-config.xml then immediately init() method on ActionServelt object will instantiate the Plug-In classes and will execute the Lifecycle methods of Plug-In class

Init(as,mc);

Destroy();

Struts.xml

ActionServlet

Plug-In

So if we write any code inside init(as, mc) method of Plug-In class then it will executed once ActionServlet init() method executes and destroy() method of plug in class will be executed once the detroy() method of ActionServlet is executed.

Using struts we can provide the support for View and Controller part of Model-View-Controller MVC architectural Design Pattern. Using our Plug in class we need to instantiate spring Class and EJBs for further operations. If our model part is deployed into EJBs. We have deploy into directory or naming services jndi services in Weblogic. In JNDI we have to bind our EJB session objects. Our struts Controller class needs to lookup the session objects. Instead of doing this we can write a plug in class called EJBStrutsPlugin classes in which we will write the codes for lookup the objects using JNDI services and then we have to communicate with EJBS using Controller classes.

In case if Model part is implemented using Spring framework. Here in need to write a Plug in classes called SpriingStrutsPlugin class and this class we have start our IOC, Using that IOC reference we need to communicate with Action Classes.

Tiles-Support

If we have a header.html, body.html, footer.html. menu.html and we want to place all the html in single jsp page then we can achieve this by sung following tags

<jsp:include page=”header.html”/>

<jsp:include page=”body.html”/>

<jsp:include page=”footer.html”/>

<jsp:include page=”menu.html”/>

Such type of composite views we can design using Tiles

We need to write a tile.xml file in which we need to configure the logical name of various views used in jsp page with the View name

**Custom RequestProcessor**

In RequestProcessor init() method will be invoked once the ActionServlet class instantiates the RequestProcessor class which is a singleton class which will provide ModuleConfig object to be passed as paraqmeter to RequestProcessor init() method. When a form is submitted by User then process() mthod on ActionServlet class will invoke the process(rq, resp) of RequestProcessor which will agail delegate the task to boolean processpreprocess(req, resp) which return is a Boolean if it returns true the it will process() method will call processLocale(req, resp) which check the browser language will store that brower at session scope and if we want to change the browser language we need to override this code. We have one more method processValidate(req,resp) and processAction(resp, req). We can override the processpreprocess() method on RerqustProcessor to write our own cutom CustomeRequestPRocessor class where we can write security related code. If security code executed successfully then return true otherwise false when it returns true to the process() method of RequestPRocessor then only it calls other methods. Using controller tag of struts-config.xml we can configure our custom RequestProcessor

**Struts 2x**

Two organizations designed struts 2sx namely opensymphony xwrok and apache Foundation and released in 2007. It a JSP model 3 architecture. Every input request to a Front Controller that should be filter class unlike servlet in struts 1.x. There are two filters ie **(i) FilterDispatcher, (ii) StrutsPrepareAndExecute**

**FilterDispacther** is used for versions previous than **2.1.3** and version **2.1.3** and newer versions we need to use **StrutsPrepareAndExecute.**

These are new generation framework

StrutsPrepateAndExecute provides following support

1. Form Backup Support
2. Input Validaions
3. Actions Support
4. Interceptors
5. AJAX Dojo’s
6. OGNL
7. Tiles Integration Support
8. Value Stacks

Struts 2 Architecture

ActionMapper

FilterDispatcher/StrutsPrepateAndExecute

Input Forms

Read

Configuration Manager

Output page

Action Proxy

read

Struts.xml

Interceptor

Action Class

Whenever users submits the form then FilterDispatcher or StrutsPrepareAndExecute FrontControllers will call ActionMapper which will read struts.xml file where configurations are written. The role of ActionMapper is to map appropriate controller class to specific request and will suggest the FilerDispatcher or StrutsPrepareAndExecute Filter class the required Controller and will contact a ActionProxy and it will add additional things and then Action Proxy reading the struts.xml file data and will get the Action Class to be executed for the input request if there is no interceptor otherwise it will execute the interceptor before the Action class and if Interceptor class ran successfully and gives a nod to execute the Action Class then Action class will executed. ActionMapper and ActionProxy classes will read the struts.xml file.

Here For validations, controller operations, multibutton and multiform operations, we have to use **ActionSupport** classes only unlike struts 1 where we have to use ActionForm or Validator Form to validate our form inputs programmatically and by xml validations respectively. In Action class only we have write validations and Controller operations.

Here default location for struts.xml config file is classes folder which in eclipse is src folder and StrutsPrepareAndExecuteFilter and we need not to give provide the location of struts.xml file in <init-param> tag of <servlet> tag while providing the configuraions for Front Controller.

In struts 2 we have been provided three types of validations approach

1. Programmatic Approach by overriding the validate() method
2. Declarative Approach using xmls by creating a xml with name beanname\_validations.xml where input validations are written
3. Annotations Approach: by using @validation annotation just before the setters of properties. Various annotation are:
   1. @RequiredStringValidator(Key=”msg”)
   2. @Email Email Validation

Controller in struts for multi button Action. If our applications is contains a single button function then we extend our controller class ActionSupport class only and if we want to write own custom name instead of execute() inside controller class then we have to configure our method name with action name ins struts.xml file.

<struts>

<constant name=*"struts.custom.i18n.resources"* value=*"resources/Messages"*/>

<package name=*"controller"* extends=*"struts-default"*>

<action name=*"reg"* class=*"controller.EmpController"* method=*"register"*>

<result name=*"input"*>/index.jsp</result>

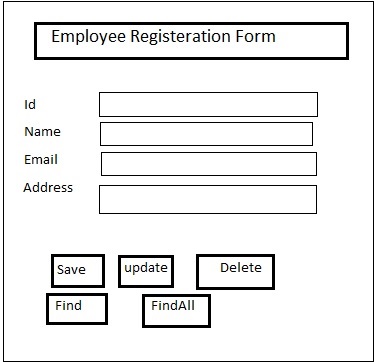
<result name=*"success"*>/success.jsp</result>

</action>

</package>

</struts>

For MultiButton operation we have to use same controller class called ActionSupport and we have to write the methods of our own name to perform various action on the click of button.



For above form we need to write the submit buttons as follows

<s:submit value=”Save” action=”savereg”>

<s:submit value=”Udpate” action=”updatereg”>

<s:submit value=”Delete” action=”deletereg”>

<s:submit value=”Find” action=”findreg”>

<s:submit value=”FindAll” action=”findAllreg”>

In struts.xml file we have to do the configurations as

<struts>

<constant name=*"struts.custom.i18n.resources"* value=*"resources/Messages"*/>

<package name=*"controller"* extends=*"struts-default"*>

<action name=*"\*reg"* class=*"controller.EmpController"* method=*"{1}"*>

<result name=*"input"*>/index.jsp</result>

<result name=*"success"*>/success.jsp</result>

</action>

</package>

</struts>

In above struts.xml file action name has been put with \* ahead of reg where \* means it’s an prefix and it will take the value from method value which will take the method name from Controller class and if action name matches with that of button then that specific method is executed. This struts.xml file also support I18N

**AJAX Tags**: Using AJAX tags we can have following things on form using DOJO Tags

1. **Tabbed panels**
2. **Dynamic AutoCompleters**
3. **DateTime picker**
4. **Dynamic Text to Format Text**
5. **Tree Tags**

We have to import DOJO library into page by using following tags

<%@taglib uri=”struts-dojo-tags” prefix=”ajax”>

Following is page for creating a tabbed panel

<%@taglib uri=*"/struts-dojo-tags"* prefix=*"sajax"*%>

<sajax:head/>

<sajax:tabbedpanel id=*"main"* closeButton=*"true"*>

<sajax:div label=*"HOME"* closable=*"true"*>

<pre>

HOMEHOMEHOMEHOMEHOMEHOME

HOMEHOMEHOMEHOMEHOMEHOME

HOMEHOMEHOMEHOMEHOMEHOME

HOMEHOMEHOMEHOMEHOMEHOME

HOMEHOMEHOMEHOMEHOMEHOME

</pre>

</sajax:div>

<sajax:div label=*"ABOUT"* closable=*"true"*>

<pre>

ABOUT ABOUT ABOUT ABOUT

ABOUT ABOUT ABOUT ABOUT

ABOUT ABOUT ABOUT ABOUT

ABOUT ABOUT ABOUT ABOUT

ABOUT ABOUT ABOUT ABOUT

</pre>

</sajax:div>

<sajax:div label=*"CONTACT"* closable=*"true"*>

<pre>

CONTACT CONTACT CONTACT CONTACT

CONTACT CONTACT CONTACT CONTACT

CONTACT CONTACT CONTACT CONTACT

CONTACT CONTACT CONTACT CONTACT

CONTACT CONTACT CONTACT CONTACT

</pre>

</sajax:div>

</sajax:tabbedpanel>

Autocomplete tags and datepicker example

<%@taglib uri=*"/struts-dojo-tags"* prefix=*"sajax"*%>

<sajax:head/>

<sajax:tabbedpanel id=*"main"*>

<sajax:div label=*"Registeration"*>

<table>

<tr>

<td>

<sajax:autocompleter label=*"Select State"* list=*"{'Andhra Pradesh','Telangana','Goa','Maharashtra','Uttar Pradesh'}"* name=*"state"*></sajax:autocompleter>

</td>

</tr>

<tr>

<td>

<sajax:datetimepicker label=*"DATE"* displayFormat=*"dd-MMM-yyyy"*/>

</td>

</tr>

</table>

</sajax:div>

<sajax:div label=*"Address"*>

<sajax:textarea labelposition=*"center"*/>

</sajax:div>

</sajax:tabbedpanel>

Textarea tag example

<%@taglib uri=*"/struts-dojo-tags"* prefix=*"sajax"*%>

<sajax:head/>

<sajax:tabbedpanel id=*"main"*>

<sajax:div label=*"Address"*>

<sajax:textarea labelposition=*"center"*/>

</sajax:div>

</sajax:tabbedpanel>

TreeView Tag Example

<%@taglib uri=*"/struts-dojo-tags"* prefix=*"sajax"*%>

<sajax:head/>

<sajax:tabbedpanel id=*"main"*>

<sajax:div label=*"Address"*>

<sajax:tree label=*"Durgasoft"*>

<sajax:treenode label=*"Hyderabad"*>

<sajax:treenode label=*"SR Nagar"*>

<sajax:treenode label=*"Near\_CanaraBank"*/>

<sajax:treenode label=*"Near\_AndhraBank"*/>

</sajax:treenode>

</sajax:treenode>

<sajax:treenode label=*"Bangalore"*>

<sajax:treenode label=*"Marathali"*>

<sajax:treenode label=*"BTM"*/>

<sajax:treenode label=*"Electronic City"*/>

</sajax:treenode>

</sajax:treenode>

</sajax:tree>

</sajax:div>

</sajax:tabbedpanel>

OGNL Expression

We can access any controller data into a jsp page by using ognl expression. Our class instance level variables we can get at presentation layer. Controller class will store the instance level data into a servlet context called **ServletActionContext Area. It will store instance object, List, String, int, session object, request and response object and along with value stack object etc.** and whenever we access the data into presentation layer using OGNL expression then it will try to fetch the data from **ServletActionContext area.** This process of getting data into JSP page using OGNL expression from **ServletActionContext** Area which stores the instance object along with maintain its properties and session, request and response scope of object is known as **PULL-MVC**.

Value Stack is also a stack object is a kind of scope object like session, request object. In ValueStack we can keep the **ServletActionServlet** Area object and values as a map and can be maintained in ActionServlet area. For doing the same we have to write the following command

HttpRequest req= ServletActionConext.getServletRequest();

ValueStack vs = ServletActionContect.getValueStack(req);

Map m= new HashMap();

m.put(“key”,”value”)

vs.push(m);

If require various scope obects in our class then we can have them in our controller class as follows where ServletActionContext is a helper class

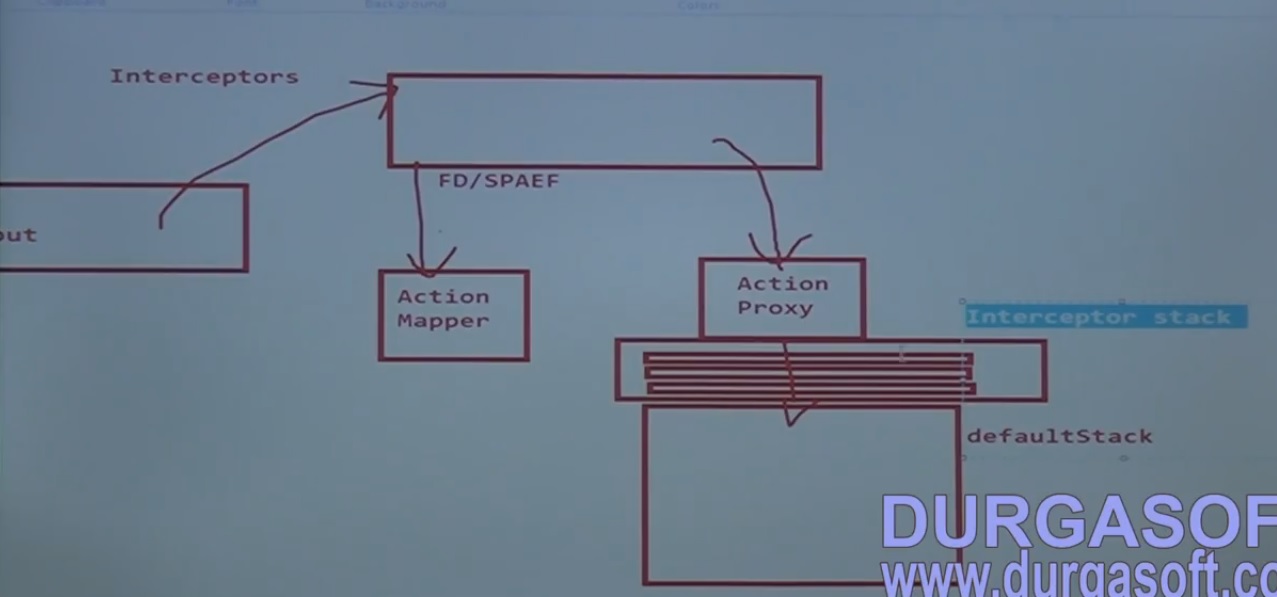
HttpRequest req= ServletActionConext.getServletRequest();

HttpSession session= ServletActionConext.getSession ();

ServletContext req= ServletActionConext.getServletContext();

Here ServletActionServlet Class is factory class and using that factory class we can each if these objects.

Intercepors



To write our own Interceptor we need to implement our interceptor class with interface called **Interceptor**  provided by struts 2.

Example of an interceptor

package controller;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import com.opensymphony.xwork2.ActionInvocation;

import com.opensymphony.xwork2.interceptor.Interceptor;

import com.opensymphony.xwork2.util.logging.LoggerFactory;

public class MyLogInInterceptor implements Interceptor {

@Override

public void init() {

System.out.println("init");

}

@Override

public String intercept(ActionInvocation ai) throws Exception {

Log l=LogFactory.getLog(HelloController.class);

l.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*in time\*\*\*\*\*\*\*\*\*\*\*");

String res=ai.invoke();

l.info("\*\*\*\*\*\*\*\*\*\*\*\*\*\*out time\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

return res;

}

@Override

public void destroy() {

System.out.println("destroy");

}

}