**Example of Complex Validations**

import javax.servlet.http.\*;

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

import net.thinksquared.registration.data.User;

public class RegistrationAction extends Action{

public ActionForward execute(ActionMapping mapping,

ActionForm form,

HttpServletRequest request,

HttpServletResponse response){

//get userid and password

RegistrationForm rForm = (RegistrationForm) form;

String userid = rForm.getUserId();

String password = rForm.getPassword();

//complex validation: check if userid exists

if(User.exists(userid)){

ActionMessages errors = new ActionMessages();

errors.add("userid", new ActionMessage("reg.error.userid.exists"));

saveErrors(request,errors);

//navigation: redisplay the user form.

return mapping.getInputForward();

}

else{

//data transformation: save the userid and password

//to database:

User user = new User();

user.setUserId(userid);

user.setPassword(password);

user.save();

//navigation: display "you're registered" page

return mapping.findForward("success");

}

}

}

The User.exists() function checks if the user ID given exists. It’s a “well-defined” interfaces between the Controller (RegistrationAction) and Model (User class).

**Detailed Internal Architecture of Struts Framework**

(login.jsp, reg.jsp) ActionServlet

Struts-config.xml

doGet(req, res)

doPost(req, res)

Init(ServletConfig sc)

String xml=sc.getInitParameter();

process(req, res) Unmarshaling

SAX

session scope RequestProcessor rp=RequestProcessorUtil(mc) ModuleConfig Object

rp.process(req, res)

Bean

RequestProcessor

processValidation(req, res)

process(req, res)

FormBeans(req, res)

Processpreprocess(req, res)

session read

processLocale(req, res)

beans typecast

User makes request using a webpage say login.jsp It will call doGet() or doGet() method. Before calling these methods it will call init(servletConfig sc) method. Using web.xml It will read init-parameter using String xml=sc.getInitParameter() and will store path of xml file ie /WEB-INF/struts-config.xml. SAX parser reads the xml data and will do marshaling ie will convert xml file to its corresponding java object of type ModuleConfig class which is a bean Class with private variables and public getter and setters method. SAX parser will store the equivalent java code into ModuleConfig Bean. It executes at deployment or at first user request.

After init(req, res) method doGet(req, rsp) or doPost Method is called. These methods will redirect the request to process(req, res) method which is a user defined method. This process() method will create the Bean Object of type ActionForm where we can store the data entered by the user. How process creates the bean object?? It will create the bean object by reading the action-mapping tag of the struts-config.xml. For the /login action of the login.jsp the reference bean is set into name attribute of action-mapping and configuration of this bean reference is done in form-bean tag under form-beans tag. Reading that configuration It will create the bean object. Under action-mapping we have an attribute called scope which is default set to session and hence the bean object is also stored in the session scope.

Now It has to run validations which is not done by the ActionServlet. For that we have another class called requestProcessor class which is a Helper Class for ActionServlet. In it also we have process(req, res) method. Mclahan requestProcessor class such that object of this class is singleton object ie only one instance of such class will be allowed. So by passing ModuleConfig object created by SAX parser, as parameter to requestProcessorUtil(mc), the single instance of requestProcessor() class created as follows:

RequestProcessor rp=RequestProcessorUtil(sc)

ActionServlet will use one requestProcessor for each module(like we have modules login.jsp, register.jsp). It will create the requestProcessor Object for one time only. After that ActionServlet uses that object only if user hits millions time. After creation of requestProcessor Object it will call process() method and delegate the same to process() method of requestProcess class.

Various features provided in struts is done in RequestProcess class only. Here we have processpreprocess(req, res) method which does preprocess tasks which include security like things which has Boolean return type. If it returns true then process() method of requestProcess will call one more method called processLocale(req, res) method and by using String lang=req.getHeaders(“accept-language”) method it will read our browser Language and will create a locale object as follows

Locale l=Locale(lang);

And will store this locale using session object

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

processLocale(req, res){ this code runs for each

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

Locale l=new Locale(lang);

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

}

After all this the process of requestProcessor will call processFormBeans(req, res) and it will read the data from req object and will store in the Bean Object. While storing data from req object to Bean Object various typecasting is done internally to avoid data mismatch exceptions.

Now the process method of requestProcess will call processValidation(req, res) and it will get the bean reference from the same session. On that bean it will call validate() method and will return the ActionError Object and then it will count the ActionErrors. If it is not equal to zero then there are some errors and same error has be displayed to the user on the same using ReqestDispatcher.forward(req, res) method. This done by bean to form configuration in the action mapping. If it finds the valid data in the bean then it will populate the valid data in the form and if it has any invalid data the same is deleted. Along with that our ActionError object data will also be populated here.