**Assignment No: 08**

**TITLE**

Design and develop any web application using Struts framework.

**OBJECTIVES**

1. To impart the efficient and available client side and server side technologies.

2. To implement the communication between computing nodes using client side and server side technologies.

3. To design and implement the web services with content management.

**PROBLEM STATEMENT**

Create a login module for the web application using struts framework.

**OUTCOMES**

*Students should be able to,*

1. Implement the effective client side and server side technologies using struts framework.

2. Solve the complex problem of development using MVC framework.

**SOFTWARE & HARDWARE REQUIREMENTS**

Software’s: Java 1.7 or Higher, Apache Tomcat 7 or higher, Struts API’s, Eclipse IDE.

**THEORY**

The frameworks plays a vital role in industries for manageable and well designed application development as well as enterprise application development. The core of the Struts framework is a flexible control layer based on standard technologies like Java Servlets, JavaBeans, Resource Bundles, and XML, as well as various Jakarta Commons packages. Struts encourages application architectures based on the

Model 2 approach, a variation of the classic Model-View-Controller(MVC)

Struts gives its own particular Controller segment and incorporates with different advancements to give the Model and the View. For the Model, Struts can collaborate with standard information get to advances, as JDBC and EJB, and also most any outsider bundles, as Hibernate, iBATIS, or Object Relational Bridge. For the View, Struts functions admirably with Java Server Pages, including JSTL and JSF, and in addition Velocity Templates, XSLT, and other introduction frameworks.

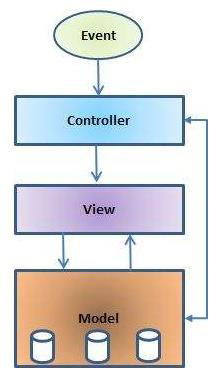
The Struts system gives the undetectable underpinnings each expert web application needs to survive. Struts causes you make an extensible advancement condition for your application, in view of distributed guidelines and demonstrated outline designs.

ORM remains for Object/Relational mapping. It is the customized and translucent constancy of items in a Java application in to the tables of a social database utilizing the metadata that portrays the mapping between the articles and the database. It works by changing the information starting with one portrayal then onto the next.

**The Model-View-Controller Architecture**

"Model-View-Controller" is a way to build applications that promotes complete separation between business logic and presentation. It is not specific to web applications, or Java, or J2EE

(it predates all of these by many years), but it can be applied to building J2EE web applications.



**Figure.5: Basic MVC Architecture.**

The "view" is the user interface, the screens that the end user of the application actually sees and interacts with. In a J2EE web application, views are JSP files. For collecting user input, you will have a JSP that generates an HTML page that contains one or more HTML forms. For displaying output (like a report), you will have a JSP generates an HTML page that probably contains one or more HTML tables. Each of these is a view: a way for the end user to interact with the system, putting data in, and getting data out.

**What is Struts?**

Struts is a framework that advances the utilization of the Model-View-Controller engineering for planning substantial scale applications. The structure incorporates an arrangement of custom label libraries and their related Java classes, alongside different utility classes. The most intense part of the Struts system is its help for making and preparing electronic structures. We will perceive how this functions later in this section.

**Struts Tags**

**Common Attributes**

Almost all tags provided by the Struts framework use the following attributes:

|  |  |
| --- | --- |
| Attribute | Used for |
| Id | the name of a bean for temporary use by the tag |
| name | the name of a pre-existing bean for use with the tag |
| property | the property of the bean named in the name attribute for use with the tag |
| scope | the scope to search for the bean named in the name attribute |

**Table.2: Struts Framework Attributes**

**Creating Beans**

Beans are created by Java code or tags.

Here is an example of bean creation with Java code:

// Creating a Plumber bean in the request scope Plumber aPlumber = new Plumber(); request.setAttribute("plumber", aPlumber);

Beans can be created with the <jsp:useBean></jsp:useBean> tag:

<!-- If we want to do <jsp:setProperty ...></jsp:setProperty> or -->

<!-- <jsp:getProperty ... ></jsp:getProperty> -->

<!-- we first need to do a <jsp:useBean ... ></jsp:useBean> -->

<jsp:useBean id="aBean" scope="session" class="java.lang.String">

creating/using a bean in session scope of type java.lang.String

</jsp:useBean>

Most useful is the creation of beans with Struts tags:

<!-- Constant string bean -->

<bean:define id="greenBean" value="Here is a new constant string bean; pun intended."/>

<!-- Copying an already existent bean, frijole, to a new bean, lima -->

<bean:define id="lima" name="frijole"/>

<!-- Copying an already existent bean, while specifying the class -->

<bean:define id="lima" name="frijole" class="com.SomePackageName.Beans.LimaBean"/>

<!-- Copying a bean property to a different scope -->

<bean:define id="goo" name="foo" property="geeWhiz" scope="request" toScope="application"/>

**Other Bean Tags**

The Struts framework provides other tags for dealing with issues concerning copying cookies, request headers, JSP implicity defined objects, request parameters, web application resources,

Struts configuration objects, and including the dynamic response data from an action. These tags are not discussed here, but it is important to be aware of their existence.

<bean:cookie ... >

<bean:header ... >

<bean:page ... >

<bean:parameter ... >

<bean:header ... >

<bean:resource ... >

<bean:struts ... >

**Bean Output**

The <bean:message> and <bean:write> tags from the Struts framework will write bean and aplication resources properties into the current HttpResponse object.

This tag allows locale specific messages to be displayed by looking up the message in the application resources .properties file.

<bean:message ... >

<bean:write ... >

<!-- looks up the error.divisionByZero resource -->

<!-- and writes it to the HttpResponse object -->

<bean:message key="error.divisionByZero"/>

<!-- looks up the prompt.name resource -->

<!-- and writes it to the HttpResponse object; -->

<!-- failing that, it writes the string -->

<!-- contained in the attribute arg0-->

<bean:message key="prompt.name" arg0='Enter a name:'/>

This tag writes the string equivalent of the specified bean or bean property to the current HttpResponse object.

<!-- writes the value of customer.getStreetAddress().toString() -->

<!-- to the HttpResponse object -->

<bean:write name="customer" property="streetAddress"/>

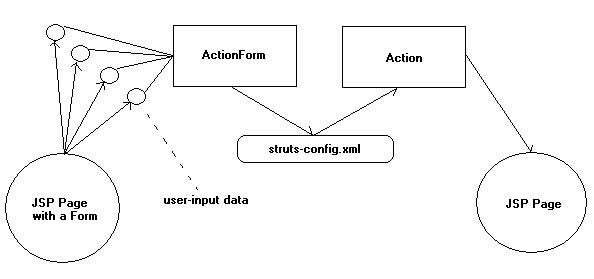
**Creating HTML Forms**

Frequently data should be gathered from a client and handled. Without the capacity to gather client input, a web application would be futile. So as to get the clients data, a html shape is utilized. Client information can originate from a few gadgets, for example, content fields, content

boxes, check takes care of, pop menus, and radio catches. The information comparing to the

client input is put away in an ActionForm class. A design document called struts-config.xml is utilized to characterize precisely how the client input are handled. The following diagram

roughly depicts the use of Struts for using forms.



**Figure.6: Struts for Using Forms**

The Struts html tags are used to generate the widgets in the html that will be used in gathering the user’s data. There are also tags to create a form element, html body elements, links, images, and other common html elements as well as displaying errors. Below are the tags provided by html section of the Struts framework and a short description of each.

<html:base> Generates a <base> tag. This tag should be used inside of a <head>

tag.

<html:button> Generates an <input type="button"> tag. This tag should be used inside a <form> element.

<html:cancel> Generates an <input type="submit"> tag and causes the Action servlet not to invoke its validate() method. This tag should be used inside a

<form> element.

<html:checkbox>

<html:multibox>

Wheat Wood Clay

Stone Sheep

<html:checkbox> Generates an <input type="checkbox">.

<html:multibox> Generates an <input type="checkbox">. "Checkednessepends upon whether the property array specified contains a corresponding value as the one specified for the multibox.

<html:errors> Generates html to display any errors that may have occurred during invocation of the validate() method.

<html:file>

<html:form> Generates <form>.

*There is a hidden element here which is invisible.* :-)

<html:hidden>

Generates <input type="hidden">.

**The ActionForm class**

The purpose of the ActionForm class is to contain and provide validation of the user-input data. This class is subclassed for application specific customization.

Here is a template for a customized ActionForm class with markers denoting where special items should be located in the class with $ symbols.

package com.akurdi.action;

import com.opensymphony.xwork2.ActionSupport;

public class LoginAction extends ActionSupport {

private String username; private String password; public String execute() {

if(this.username == null || this.password == null)

{

return "error";

}

if (this.username.equals("admin")

&& this.password.equals("admin123")) {

return "success";

} else { addActionError(getText("error.login")); return "error";

}

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword()

{

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**Do this:**

1. Create the directory structure. The root directory is SimpleStruts, and it has the standard WEB-INF directory with classes inside, and com.akurdi.action inside that. It also has a lib directory within WEB-INF, which is something we haven't seen before; we'll see in a minute what goes there.

2. Copy the Struts tag library descriptor files into WEB-INF. The files struts.tld, struts-bean.tld, struts-form.tld, struts-html.tld, struts-logic.tld, and struts-template.tld are available in the lib directory of your Struts installation.

3. Copy the Struts parser, struts.jar, into WEB-INF/lib/. This file is available in the lib directory of your Struts installation

4. Create the tag descriptor library file for any custom tags you may use beyond the Struts tags. In this case, the file defines no custom tags, but it's good practice to have it in place, in case you need to add your own tags later.

5. Create the struts-config.xml file.

There are three main sections to a struts-config.xml configuration file. They are the "Form Bean Definitions" section, the "Global Forward Definitions" section, and the "Action Mapping Definitions" section defines a forward called "success".

6. Create the web.xml file.

The web.xml web application configuration file will need to define the servlet ActionServlet, to which control will be transferred whenever an appropriate URL pattern is accessed. The servlet is defined just as any other servlet will be defined. The URL pattern is specified by a servlet mapping. For this application, the URL pattern is any requested resource that ends with a .do extension.

In order to use the Struts tags, the .tld files describing the tags will need to be included in the configuration file. The references to these tags are made just as they were for our own custom tags in the previous chapter. The Struts framework is simply a complex set of tag libraries (struts\*.tld), with associated code (struts.jar).

7. The ApplicationResources.properties file provides resources that will be used by any subclassed Struts classes (for example, SetNameAction). This resources file provides a place to define prompts, labels that will display on buttons, and other information that may change. By placing this information in the ApplicationResources.properties file, recompiling any servlets used in the application can be avoided, as well as encouraging separation of logic and presentation.

8. Instances of the Name class are placed in the user sessions. Only one will exist in any particular user session. It provides methods for accessing and mutating a name.

9. The view of the application is done with the JSP index.jsp. It represents the user interface and allows the user to interact with the application.

10. Compile, create .war file and place it into the webapps folder of tocat. *Manual*

11. Go to [http://localhost:8080/Sturtsdemo/example/Login.js](http://localhost:8080/Sturtsdemo/example/Login.jsp)p to test your application

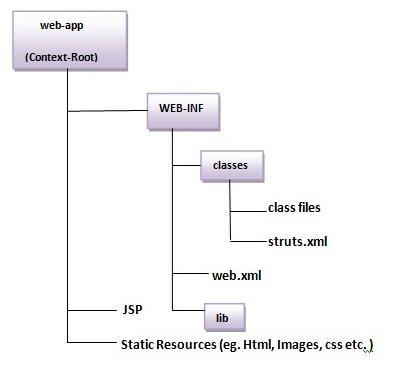
**TECHNOLOGY/TOOL**

1) Eclipse IDE

2) Apache Tomcat 7.0 or higher

**DESIGN/EXECUTION STEPS**

Step 1) Create the directory structure as



Step 2) Create input page as below.

Step 3) Provide the entry of Controller in (web.xml) file as given above

Step 4) Create the action class (LoginAction.java)

Step 5) Map the request in (struts-config.xml) file and define the view components. Step 6) Load the jar files

Step 7) start server and deploy the project or create .war file paste it in webapps folder and run from manager-app.

**TEST CASES**

1. Manual test cases need to be performed on struts application.

2. Check whether the tomcat server is running.

3. Check whether the application running status in manager app list is true.

4. Run the application by selecting the app in the list.

5. Check whether the application giving desired results.

**CONCLUSION/ANALYSIS**

Hence we have successfully tested the Struts framework and tested the results.

**PROGRAM CODE: INPUT & OUTPUT**

1) Deploy the application from tomcat manager.

2) Provide the login username and password for login.

3) Test the result for correct and incorrect credentials.

**ORAL QUESTIONS**

1) What are the components of Struts Framework?

2) What’s the role of a handler in MVC based applications?

3) What’s the flow of requests in Struts based applications?

4) Which file is used by controller to get mapping information for request routing?

5) What’s the role of Action Class in Struts?

6) How an actionForm bean is created? And Its uses.

7) How validation is performed in struts application?

8) What’s the purpose of Execute method of action class?

9) How can we display all validation errors to user on JSP page?

10) What are the benefits of Struts framework?