Fundamentos para el desarrollo de aplicaciones en la red

Tema: Frameworks de desarrollo sobre páginas JSP

Autor: Elías Niño

JSF

Uno de los tantos frameworks de desarrollo es conocido como JSF (Java Server Faces) el cual está basado en el conocido lenguaje de programación Java. JSF este tiene como fin, simplificar el desarrollo de interfaces web en las aplicaciones JEE(Java Enterprise Edition).

Como características principales de JSF podemos mencionar que este utiliza el patrón MVC (Modelo Vista Controlador) en la creación de las aplicaciones de esta forma:

- JSF Utiliza páginas JSP para generar las vistas de usuario
- Asocia cada vista que se crea a un conjunto de objetos java(clases) las cuales son conocidos como ManagedBeans estos facilitan la forma en que se manipula la información.

JSF también puede ser integrado con otro framework de desarrollo conocido como Hibernate el cual completa el modelo MVC añadiendo una nueva capa, la capa de Persistencia.

¿Cómo funciona JSF?

Normalmente una página web html consta de etiquetas las cuales brindan elementos para la navegación en un portal, estas etiquetas sirven para crear/mostrar botones, textos, imágenes, listas, entre otros.

La principal función del controlador JSF es asociar a cada página JSF una clase Java y desde allí manejar la información que el usuario digita. Utilizando JSF le simplifica al programador la forma en que se muestran datos al usuario, leer datos que el usuario escribe, controlar los estados, entre otros.

Además de las páginas JSP y su asociación con un archivo.java , este también posee Beans Java los cuales se conectan a los formularios JSF, además de esto JSF como cualquier framework posee archivos de configuración.

Etiquetas JSF

El framework JSF trae consigo tags personalizados los cuales se utilizan para el desarrollo de aplicaciones, a continuación se mostrarán algunos de estos tags con sus ejemplos respectivos:

FORM

Etiqueta JSF:

```
<h:form id = "nombreID">
    ....
</h:form>
```

Salida HTML:

```
<form id = "nombreID">
    ....
</form>
```

Descripción:

Como bien se ve en el ejemplo esta etiqueta crea la misma etiqueta HTML form, cabe resaltar que dependiendo de la IDE que se utilice para insertar estas etiquetas esta le agregará otros parámetros por ejemplo si utilizamos Netbeans, este le agregará parámetros como **name, method, action, enctype**.

COMMANDBUTTON

Etiqueta JSF:

```
<h:commandButton id = "nombreID" value = "Nombre"></h:commandButton>
```

Salida HTML:

```
<input id = "nombreID" type = "submit" name = "nombreID" value = "Nombre">
```

Descripción:

La etiqueta commandbutton se utiliza para agregar botones en la página, estos tienen las mismas propiedades que los creados por HTML, si se utiliza una IDE como Netbeans podremos ver utilizando las teclas Ctrl + Espacio cuales son las propiedades que tiene el elemento.

OUTPUTTEXT

Etiqueta JSF:

```
<h:outputText value = " ejemplo "/>
```

Salida HTML:

ejemplo

Descripción:

Esta etiqueta permite imprimir una cadena en la página, simplemente se indica el valor que esta va a tomar. Algo bastante útil es que se le puede indicar expresiones EL las cuales se tratarán a continuación.

OUTPUTLINK

Etiqueta JSF:

```
<h:outputLink value = "direccion" >
    <h:outputText value = "Link" />
</h:outputLink>
```

Salida HTML:

```
<a href = "direccion" > Link </a>
```

Descripción:

La etiqueta outputLink es utilizada para la creación de hipervínculos, en el lenguaje HTML se utilzaría la etiqueta <a … > como se muestra en el ejemplo, se pueden insertar etiquetas dentro del tag, cabe recalcar que estas etiquetas pueden utilizar el lenguaje EL.

GRAPHICIMAGE

Etiqueta JSF:

```
<h:graphicImage id = "imagenID" alt = "texto" url = "imagen.jpg">
</h:graphicImage>
```

Salida HTML:

```
<img id = “imagenID” src = “imagen.jpg” alt = “texto” />
```

Descripción:

La etiqueta graphicImage nos sirve para insertar una imagen en la página jsp, este tag jsf genera el mismo código que la etiqueta en HTML, en el ejemplo anterior se carga una imagen .jpg y se le especifica un texto de descripción de la imagen.

INPUTTEXT, INPUTSECRET Y INPUTTEXTAREA

Etiquetas JSF:

```
<h:inputText id = “idTexto” value = “tx1” />

<h:inputSecret id = “idSecreto” redisplay = “false” />

<h:inputTextarea id = “idTA” rows = “4” cols = “7” value = “tx3” />
```

Salida HTML:

```
<input id = “texto” type = “text” name = “idTexto” value = “tx1” />

<input id="idSecreto" type="password" name="idSecreto" value="" />

<textarea id="idTA" name="idTA" cols="7" rows="4">tx3</textarea>
```

Descripción:

La estas etiquetas nos sirven para la captura de la información, podemos encontrar el campo tipo “text” el cual sirve para entrada de datos normales, el tipo “password” para campos de contraseña o que no deben ser vistos y por último el

textarea, nos damos cuenta que la sintaxis es muy parecida a la sintaxis HTML y que tiene las mismas propiedades aunque algunas tienen variaciones de sus nombres.

SELECTBOOLEANCHECKBOX

Etiqueta JSF:

```
<h:selectBooleanCheckbox title="titulo" value="valor" >
</h:selectBooleanCheckbox>

<h:outputText value="Desea usted ir de viaje?"/>
```

Salida HTML:

```
<input type="checkbox" title="titulo" />Desea usted ir de viaje?
```

Descripción:

Como bien podemos ver, esta etiqueta sirve para la creación de etiquetas tipo “checkbox”.

A partir de ahora las etiquetas cambian un poco ya que utilizan el lenguaje EL, el cual permite que un objeto visual obtenga valores declarados en alguna clase, veamos algunos ejemplos.

Salida HTML:

```
<span id="jsftags:carros">
  <table>
    <tr>
      <td><label for="jsftags:carros">
          <input name="jsftags:carros" value="1" type="checkbox">Opcion 1</input>
      </label></td>
      <td><label for="jsftags:carros">
          <input name="jsftags:carros" value="2" type="checkbox">Opcion 2</input>
      </label></td>
      <td><label for="jsftags:carros">
          <input name="jsftags:carros" value="3" type="checkbox">Opcion 3</input>
      </label></td>
    </tr>
  </table>
</span>
```

SELECTMANYCHECKBOXLIST

Descripción:

Como vemos en el ejemplo el simple hecho de configurar parte del código java hemos obtenido varios checkbox automáticos, más adelante se hablara de las ventajas que trae utilizar el lenguaje EL.

SELECTONEMENU

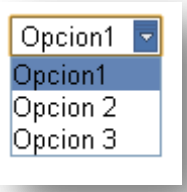
Etiqueta JSF:

```
<h:selectOneMenu id="idS" value="#{Bean.propiedad}">
<f:selectItems   value="#{Bean.lista }" />
</h:selectOneMenu>
```

Salida HTML:

```
<select id="idS" name="nombreS" size="1">
<option value="1">Opcion 1</option>
<option value="2"> Opcion 2</option>
<option value="2"> Opcion 3</option>
</select>
```

Vista del elemento:



Descripción:

Esta etiqueta nos sirve para crear un menú de selección única el cual posee diferentes opciones, en el ejemplo: Opcion 1, Opcion 2 u Opcion 3.

Como podemos ver existen muchas etiquetas JSF que nos ayudan a ahorrar tiempo al momento de crear la Vista del usuario, claro está que todo esto es debido a una correcta organización y buenas prácticas de programación como lo es utilizar el patrón MVC de una forma correcta.

Backbeans

A cada clase java que se le ha asociado un formulario (página) JSF, se le conoce como BackEnd Beans ya que son clases java que se encuentran por detrás del formulario, aquí es donde se lleva a cabo toda la lógica de la página, esto nos resuelve el problema que ocurre en páginas JSP en las cuales el código se encuentra mezclado con la vista lo cual dificulta el mantenimiento y el entendimiento del código.

Recuerda: La estructura de una página JSF consiste en una página JSP y un Backbean (clase .java).

El controlador JSF regista en el servidor de aplicaciones la extensión .jsf que estará asociado a este tipo de páginas. Los pasos cuando se carga una página son los siguientes, primero se comprueba si es la primera vez que se ingresa a la página, si es cierto, se carga la página jsp asociada y se procesa construyendo en memoria la representacion de los controles de la página. Luego de esto JSF sabe como construir el código html el cual será la salida, es decir este sabe lo que contiene y como renderizarla.

En el segundo paso se le asocia el backbeans que este contiene, normalmente este backbean debe llamarse como la página original pero con extensión .java. Para esta asociación, el controlador ha obtenido la lista de todos los backbeans asociados y de esta manera los busca en los ámbitos de la aplicacion (request y session).

Eltercer y último paso consiste en dar valores a las propiedades de los elementos JSF de la página (Lenguaje JSF o más conocido como EL).

Navegación entre páginas

JSF posee un mecanismo de navegación algo particular, en el luego de haberse ejecutado una acción el controlador determina como se debe mostrar al usuario el resultado de su peticion, existen varias posibilidades las cuales son:

- Terminar la petición mostrando la misma página que la invocó
- Llevar a otra página diferente
- Enviar la peticion de redirección

Este mecanismo de navegación se implementa de una manera muy simple en una pagina JSF. Cuando el controlador JSF invoca al método este devuelve un String el cual indica a que página se llevara. Si este es nulo, se irá a la misma página, si posee un valor se irá a ese valor el cual se ha especificado en los archivos XML de configuración o reglas, estas reglas pueden definir:

- La página de origen (quien originó la petición)
- Etiqueta de destino.
- La página de destino para cada etiqueta. Es el JSP en el que se procesará la petición de salida la cual utiliza el request y la variable de sesión.

En el siguiente código vemos un ejemplo de un archivo .XML de configuración para navegación entre páginas el cual fue generado en NetBeans:

```
<faces-config version="1.2"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd">
    <navigation-rule>
        <from-view-id>/paginaPrimera.jsp</from-view-id>
        <navigation-case>
```

```
            <from-outcome>irPagina1</from-outcome>
            <to-view-id>/paginaSegunda.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
    <navigation-rule>
        <from-view-id>/paginaSegunda.jsp</from-view-id>
        <navigation-case>
            <from-outcome>irPagina2</from-outcome>
            <to-view-id>/paginaPrimera.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
</faces-config>
```

En este código podemos fácilmente ver la navegación entre páginas, vemos que la página llamada “paginaPrimera” puede comunciarse con la página llamada “paginaSegunda” y que el proceso también funciona de la segunda a la primera.

Muchos IDE permiten realizar este tipo de configuraciones de navegación de una forma gráfica como lo es Netbeans, lo cual hace bastante fácil la creación del código, cabe resaltar que Netbeans también permite que el usuario escriba su propio código XML.

Procesamiento de peticiones

Existen diferentes etapas en el procesamiento de una petición estas se encuentran definidas dependiendo de la versión e JSF, esto implica que en un futuro pueden ser otras. El procesamiento de las peticiones tiene un vínculo directo con el ciclo de vida, veamos cuales son:

- Restaurar los componentes de la vista (Restore View): Es la etapa encargada de construir la estructura de los componentes en memoria.

- Aplicar los valores de la petición (Apply Request Values): En esta etapa se recupera el valor del request y se le asignan los beans de la página.

- Procesamiento de las validaciones (Process Validations). Se realiza la verificación de los parámetros de entrada según reglas definidas.

- Actualizar valores del modelo(Update Model Values). Los valores que anterior mente fueron leídos y se revisaron según las reglas, son cargados en cada Bean.

- Invocación (Invoke Application): Se ejecutan las acciones y eventos especificados de la página.

- Generación de la página (Render Response): En esta última etapa se renderiza(crea cada componente visual que será mostrado al usuario) la página y se le envía al usuario.

Lenguaje EL

Es el lenguaje creado por JSF para la recolección y trato de la información, con este lenguaje es posible realizar lo que anteriormente vimos al describir alguno de los tags JSF, con este lenguaje es muy sencillo acceder a la información o propiedades de los Beans que hayamos creado.

La sintaxis de este lenguaje se define de la siguiente manera:

#{MiBean.propiedad}

Como bien hemos visto esto permite que podamos ingresar o leer valores dependiendo de nuestras necesidades.

Veamos algunos ejemplos:

Ejemplo EL	Tipo	Resultado
MiBean.elementostring	String	Se muestra el valor del String
MiBean.elementoBoolean	Boolean	Al igual que el objeto Boolean este devuelve *True* o *False*
MiBean.Propiedad1.Propiedad2	Propiedad: Clase, Propiedad2: String	Devuelve el String que se encuentra dentro del objeto del vean.
MiBean.hash['valor']	Hashmap	Regresa el elemento de la clave "valor"
MiBean.hash['valor'].propiedad	Hashmap, Propiedad	Regresa la propiedad del elemento con clave "valor" del hashmap
MiBean.nombreHash.valor	Hashmap	Es otra forma de acceder a la información que se encuentra en el Hashmap, este devuelve el elemento de la clave "valor"

FacesContext

Como bien conocemos un BackBean es una clase java común y corriente como todos conocemos, esta no tiene ni conoce nada de la aplicación. El entorno JSF provee una clase la cual conoce todo el entorno de la aplicación esta es conocida como FacesContext, esta clase le da a cada Bean la posibilidad de conocer "Qué hay afuera" además de esto le permite al vean conocer el contexto HTTP. Con esto nuestro Bean tiene el acceso a otros beans y propiedades de la aplicación, puede conocer la petición HTTP que se originó, entre otros.

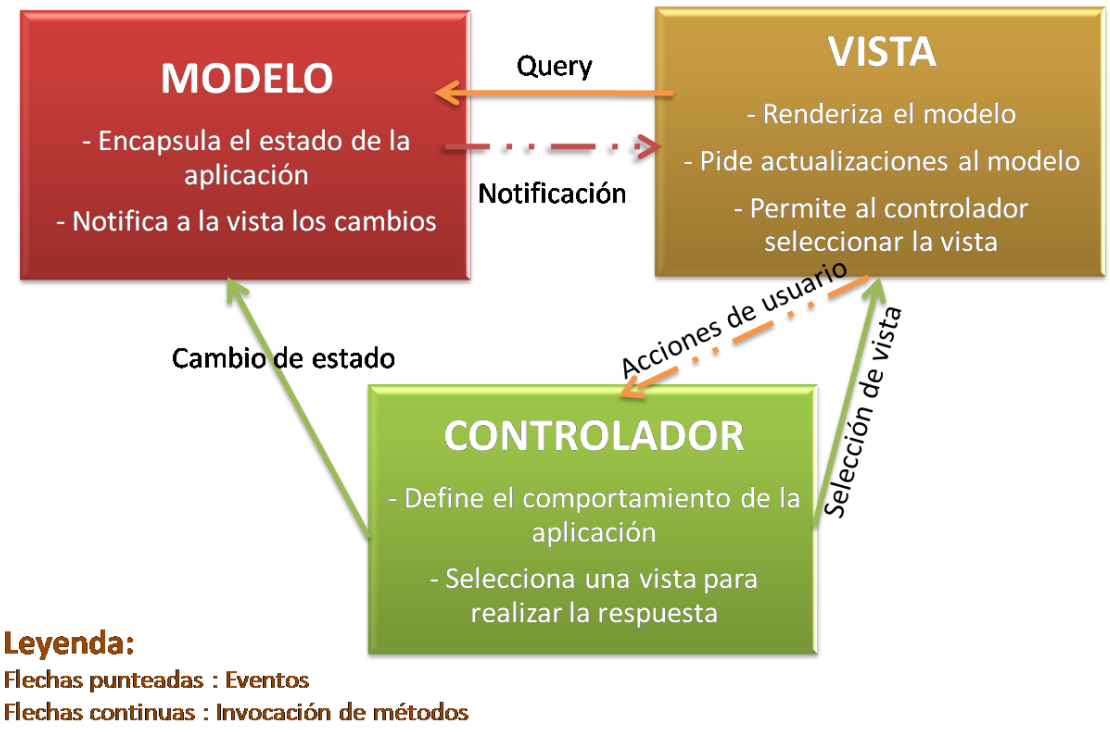
```
public void metodo1(){
        //De esta forma podemos obtener el contexto
        FacesContext fc = FacesContext.getCurrentInstance();

        //Si queremos obtener el request ...
        HttpServletRequest rq = (HttpServletRequest) fc;

        //Podemos obtener los parametros enviados por medio de:
        Enumeration e = rq.getParameterNames();

        //El ejemplo anterior es para obtener todos los elementos pero si solo queremos uno podemos invocar el método

        String nombre = rq.getParameter("nombre");

        //Si queremos acceder a la session activa utilizamos
        HttpSession s = rq.getSession();
}
```

En el ejemplo anterior se muestra alguna de las muchas utilidades que tiene el FaceContext, para más información es aconsejable revisar la API en http://java.sun.com/javaee/javaserverfaces/1.2_MR1/docs/api/index.html

APACHE STRUTS

¿Qué es Apache Struts?

Al igual que JSF, Apache Struts es un framework de desarrollo el cual está basado en el patrón MVC(Modelo Vista Controlador). Veamos un gráfico para tener mayor conocimiento acerca de este patrón de desarrollo.



FUENTE DE LA IMAGEN: http://struts.apache.org/

En el siguiente gráfico podemos ver la forma en que funciona realmente la web utilizando la tecnología de java claro está, analicemos la gráfica:

FUENTE DE LA IMAGEN: http://struts.apache.org/

Todo comienza con una solicitud que es enviada por el usuario que visita la web, esta solicitud es capturada por un Servlet (Controlador) el cual utiliza objetos como Request o response para hacer tratamiento de esta información, este Servlet puede llamar a los Beans creados en la aplicación o los creados automáticamente para aquellas personas que utilizan IDE's, en los Beans (Modelo) es en donde se hace el tratamiento de toda la información y es donde se encuentra la lógica de todo el negocio, cuando se tiene una respuesta el Servlet reenvía a la vista para que esta genere la página en que en este caso sería una página JSP para el usuario, la vista antes de generar una respuesta puede tener comunicación con el

modelo por si se necesita realizar alguna consulta, esto varía dependiendo del modelo que se utilice.

El modelo

Comenzaremos hablando sobre el papel que juega el modelo en el marco de trabajo (framework) Struts, en este componente como bien sabemos se establece la lógica del negocio para esto utilizaremos Action Beans los cuales nos permitirán tener la comunicación de la información que llega de la vista y poder realizar el procedimiento que necesitemos.

Debemos tener en cuenta que los Action Beans nos permiten tener objetos como el Request o la el objeto Session los cuales son vitales en el desarrollo de este componente.

Para poder crear el modelo es necesario tener una clase Java la cual extenderá de Action, si utilizamos una IDE como Netbeans esta clase se creará automáticamente y tendremos los métodos necesarios para definir nuestra lógica.

Veamos un código de validación de un formulario y lo simple que es realizar el procedimiento utilizando Struts:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package Modelo;

import controlador.LogonForm;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

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

public class NewStrutsAction extends org.apache.struts.action.Action {

    /* forward name="success" path="" */
    private final static String SUCCESS = "success";

    /**
     * This is the action called from the Struts framework.
     * @param mapping The ActionMapping used to select this instance.
     * @param form The optional ActionForm bean for this request.
     * @param request The HTTP Request we are processing.
     * @param response The HTTP Response we are processing.
     * @throws java.lang.Exception
     * @return
     */
    public ActionForward execute(ActionMapping mapping, ActionForm form,
            HttpServletRequest request, HttpServletResponse response)
            throws Exception {
        // Validando los parámetros

        String username = ((LogonForm) form).getUsername();
        String password = ((LogonForm) form).getPassword();

        ActionErrors ae = new ActionErrors();

        if (username.equals("")) {
            ae.add(ActionErrors.GLOBAL_ERROR, new
ActionError("error.username.blank"));
            saveErrors(request, ae);
```

```
            return (new ActionForward(mapping.getInput()));
        }

        if (password.length() < 10) {
            ae.add(ActionErrors.GLOBAL_ERROR, new
ActionError("error.password.length"));
            return (new ActionForward(mapping.getInput()));
        }

        return mapping.findForward(SUCCESS);

    }

}
```

En este sencillo ejemplo validamos dos campos simples, el usuario no podía ser diferente d enulo y la contraseña no podía tener menos de 10 caracteres, si alguno de estos dos se daba se generaba un error y se agregaba, hay algo muy importante en Struts y es el manejo de errores ya que al crear el proyecto existe un archivo el cual guarda todos los errores, estos ya se encuentran definidos por el usuario y tan solo es necesario especificar qué tipo es para que aparezca el mensaje.

System State Beans

Los System State Beans son el conjunto de objetos de negocio que representan el estado actual del sistema, por ejemplo: el carrito de la compra que el usuario va modificando a lo largo de su interacción con la aplicación. Estos objetos de negocio serán típicamente JavaBeans o EJBs de los que se guardará referencia en la sesión del usuario, que serán modificados desde los Action y que serán

consultados desde las JSPs. Esta clase de objetos no debiera tener ningún conocimiento de la View.

BusinessLogic Beans

Los objetos de negocio son los que implementan la lógica de negocio, el cómo hacer las cosas y su propia persistencia. Estos objetos de negocio no debiera tener ningún conocimiento de la View o el Controller de forma que debieran ser perfectamente reutilizables para implementar soporte a distintas interfaces y hasta para incluirse en nuevas aplicaciones.

La vista

A continuación conoceremos más acerca de la vista, esta se encuentra formada básicamente por la pagina JSP y todo aquello componente que tenga que ver con la creación de la interfaz de usuario o de su modificación.

Creación de páginas multi-idiomas

Struts provee una forma de poder mostrar el contenido de nuestras páginas web, en diferentes idiomas, para ello se debe crear un archivo en el cual se establezcan todas las traducciones o significados de cada elemento de la página un ejemplo del archivo podría ser.

…
//Archivo español
encabezado.titulo = “Este es el titulo en español”
encabezado.imagen = “Este es el src de la ruta para la imagen en español”
estructura.titlePI = “Este es el titulo del panel izquierdo”

estructural.titlePD = "Este es el título del panel derecho"

...

…

//Archivo en Ingles
header.title = "This is the title"
header.img = "This is the url of the image"
panelL.title = "This is the title of the left panel"
panelR.title = "This is the title of the right panel"
…

Para cada idioma se creara el archivo diferenciándolo solamente en el nombre, la diferencia existe en que las dos ultimas letras deben ser el código ISO del nombre un ejemplo seria idioma_xx.properties.

Luego iremos al archivo struts-config.xml y configuraremos el tag /servlet/init-param/param-name application y establecemos en el param-value la localización del archivo:

```
<servlet>
    <servlet-name>action</servlet-name>
    …
    <init-param>
        <param-name>application</param-name>
        <param-value>com.carpeta.aplicacion.myapp</param-value>
    </init-param>
…
```

Ahora en el archivo web.xml de nuestra solución debemos agregar:

<web-app>

…

<taglib>

<taglib-uri>/WEB-INF/struts-bean.ltd</taglib-uri>

<taglib-location>/WEB-INF>struts-bean-ltd></taglib-location>

</taglib>

…

</web-app>

Y ahora debemos tener en cuenta que en cada lugar que utilicemos esta característica debemos incluir

…

<&@ taglib uri = “/WEB-INF/struts-bean.ltd” prefix = “bean” %>

…

Para de esta forma decir que estamos utilizando la taglibrary struts-bean

Como último paso se utilizará el tag <vean:message key = “clave.subclave” /> dependiendo del caso en el que lo utilicemos como por ejemplo:

…

<h1><bean:message key = “encabezado.titulo”/></h1>

…

Algo para recordar es que Struts automáticamente asigna el lenguaje dependiendo del idioma principal de la aplicación.

Forms

A la hora de desarrollar aplicaciones la tarea que más tiempo consume es la de crear las vistas de usuario y los formularios de entrada/salida de datos, comprobación de errores, presentar form con los mismo valores ingresados luego que se ha equivocado, etc. Por este tipo de inconvenientes Struts ha ideado una forma de simplificar este proceso veamos un ejemplo:

- Primero debemos crear una clase que extienda de ActionForm
- Creamos la página JSP

```
...
<%@ taglib uri= "/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri= "/WEB-INF/struts-html.tld" prefix="html" %>
...
<html:html>
...
  <html:form action="/logon" focus="username">
...
    <h1><bean:message key="logon.header"/></h1>
    <html:form action="/logon" focus="username">
    <h5><html:errors/></h5>
    <h3><bean:message key="logon.mainText"/></h3>
    <p><bean:message key="logon.username"/>
        <html:text property="username" size="16" maxlength="16"/></p>
    <p><bean:message key="logon.password"/>
        <html:password property="password" size="16" maxlength="16"
redisplay=false/></p>
    <p><html:submit property="submit" value="Submit"/></p>
    <p><html:reset/></p>
```

```
...
  </html:form>
...
</html:html>
```

Luego declaramos el ActionForm en el struts-config.xml agregando en /struts-config/form-beans el tag <form-bean name="nombreForm" type="paquete.clase"/> y en la declaración del Action agregamos los atributos name="nombreForm", scope="(request ó session)", e input="paginaForm.jsp". De esta menera tendríamos:

```
<struts-config>
  ...
  <form-beans>
  ...
   <form-bean  name="logonForm" type="com.empresa.aplicacion.LogonForm"/>
  ...
  </form-beans>
  ...
  <action-mappings>
  ...
   <action          path="/logon"          type="com.empresa.aplicacion.LogonAction"
name="logonForm"
            scope="request" input="/logon.jsp">
    ...
   </action>
   ...
  <action-mappings>
  ...
</struts-config>
```

El controlador

Segun el patrón MVC el controlador maneja la funcionalidad involucrada desde que el usuario entra a la web hasta que este sale, cada click que de todo pasa por este componente. Este llama a objetos del negocio del modelo para que resuelvan lo necesario, y segun lo que suceda ejecutará la página JSP para mostrarle una salida al usuario.

Struts incluye un servlet el cual recibe solicitudes del usuario este llama al Action Bean que corresponda y segun lo que este retorne se ejecuta una página JSP. Veamos un ejemplo

Primero debemos hacer una clase que extienda de Action (org.apache.action.Action) luego vamos al archivo struts-confic.xml para agregar este nuevo mapping:

```
<struts-config>
...
  <action-mappings>
...
    <action path="/logoff" type="com.empresa.aplicacion.LogoffAction">
      <forward name="success" path="/index.jsp"/>
    </action>
...
  </action-mappings>
...
</struts-config>
```

En código anterior cuando se direccione a /logoff el controlador llamará a LogoffAction, luego de hacer las acciones respectivas si esta devuelve un forward entonces se ejecutará el forward y se re direccionará al usuario a /index.jsp. Si una

accion es asociada a un formulario se debe definir un FormBean, un Action Mapping con el Form Bean asociado y los forwards necesarios.

Ej:

```
<struts-config>
...
  <form-beans>
...
    <form-bean  name="logonForm" type="com.empresa.aplicacion.LogonForm"/>
...
  </form-beans>
...
  <global-forwards>
...
    <forward   name="success" path="/mainMenu.do"/>
...
  </global-forwards>
...
  <action-mappings>
...
    <action path="/logon" type="com.empresa.aplicacion.LogonAction"
          name="logonForm" scope="request" input="/logon.jsp"> </action>
...
  </action-mappings>
...
</struts-config>
```

En este caso hemos definido un forward global el cual es un forward que aplica a cualquier Action Mapping.

Si se desea ampliar los conocimientos sobre Apache Struts se puede visitar su página oficial en la que se encuentra amplia documentación sobre este framework, en http://struts.apache.org/

Bibliográficas

http://java.sun.com/javaee/javaserverfaces/1.2_MR1/docs/api/index.html, 7 de Marzo de 2010

http://www.horstmann.com/corejsf/jsf-tags.html, 8 de Marzo de 2010

http://www.exadel.com/tutorial/jsf/jsftags-guide.html, 8 de Marzo de 2010

http://www.roseindia.net/jsf/jsftags.shtml, 8 de Marzo de 2010

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava, 7 de Marzo 2010

http://www.programacion.com/java/tutorial/joa_struts/1/, 12 de Marzo de 2010

http://struts.apache.org/, 12 de marzo de 2010