DESARROLLO DE APLICACIONES WEB

Introducción a Struts Framework

---

1. Objetivo:

- Introducir los conceptos relacionados con la construcción de aplicaciones Web Java basadas en el Framework Web MVC2 Struts.

2. Prerrequisitos:

Esta práctica fue realizada con las siguientes versiones de Software:

- Apache Tomcat Web Server 7.0.x o superior.
- JDK 1.7.x.
- Bloc de notas.
- Struts 1.3.10 o superior.

3. Descripción:

Primeros pasos

- La primera parte es conceptual. Se deben afianzar los conceptos de Framework, Framework Web, ejemplos de algunos Frameworks Web y luego un vistazo general al Framework Struts.
- Según las definiciones que se puede leer en http://es.wikipedia.org/wiki/Framework y http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/, defina con sus propias palabras, ¿Qué es un framework?
- En el ámbito de las aplicaciones sobre Internet, la definición se especializa un poco. Según el siguiente texto http://www.cssblog.es/guias/Framework.pdf como define Usted, con sus propias palabras, ¿Qué es un framework web? ¿Cómo ayuda al desarrollo de aplicaciones web?
- Existen muchos frameworks web para diferentes lenguajes de programación. En el caso de Java, es difícil realizar una selección. De hecho la selección no se debe hacer por cual es el mejor, sino por el tipo de problema a atacar. Según estas lecturas http://orangeslate.com/2007/11/27/six-best-web-frameworks-in-java/, http://olex.openlogic.com/wazi/2010/choosing-the-right-java-web-development-framework/ se puede tener una visión un poco más amplia al respecto.

- Investigue al menos 5 frameworks web Java adicionales que no se mencionen en las direcciones anteriores y defina porque los utilizaría. Como sugerencia puedo mencionar ZK, Play y Click.

Hacia el Hola Mundo Struts

- El framework Struts es un proyecto de la Fundación Apache y toda la información relacionada con el proyecto está en su página en la siguiente dirección: http://struts.apache.org/.
- Es necesario entender de que se trata un framework antes de iniciar a conocer alguno de ellos.
- En el caso de Struts, ofrece una librería y una arquitectura de aplicación web basada en el patrón Modelo Vista Controlador (MVC) y se encuentra disponible en dos versiones, actualmente la 1.3.x y la 2.3.x. En esta ocasión se va usar la versión 1.3.10 de Struts.
- Descargue el archivo **struts-1.3.10-all.zip**, desde la siguiente dirección http://struts.apache.org/download.cgi#struts1310.
- Extraer el contenido del archivo en el disco duro, por ejemplo, en la raíz de la unidad de disco C. Se debe ver algo como lo siguiente:

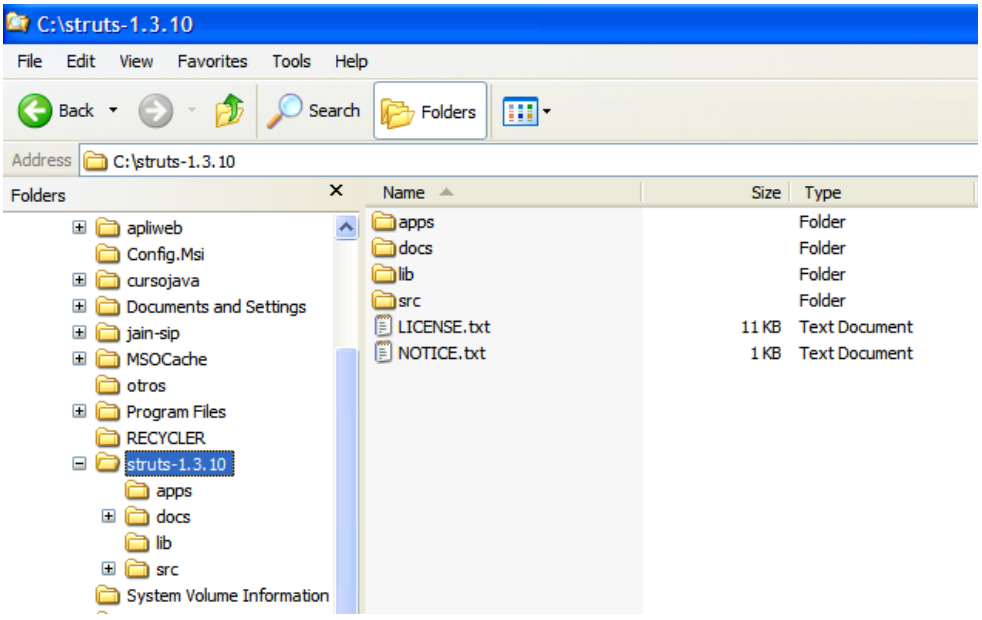

Figura 1. Contenido del archivo struts-1.3.10-all.zip

- El directorio **lib** contiene todos los archivos necesarios correspondientes a la implementación del framework Struts. El directorio **docs** contiene toda la documentación correspondiente con Struts. El directorio **apps** contiene algunos

ejemplos de aplicaciones basadas en Struts que se pueden ejecutar, como referencia, en algún servidor web Java como Tomcat, simplemente copiando el archivo war en el directorio webapps de Tomcat. El código fuente de estas aplicaciones se encuentra en **src** para que pueda ser estudiado.

- Para construir una aplicación java web usando Struts se inicia de la misma manera que para una aplicación web java tradicional. Se debe crear la estructura ya conocida y que se muestra en la figura 2. Pero dentro del directorio de trabajo de la asignatura (**apliweb**) se debe crear el directorio del proyecto llamado **mistruts** y dentro los directorios **src** y **web**. En este último (**web**) se crea la estructura antes mencionada. La estructura de directorios debe ser similar a la que se muestra en la figura 3.

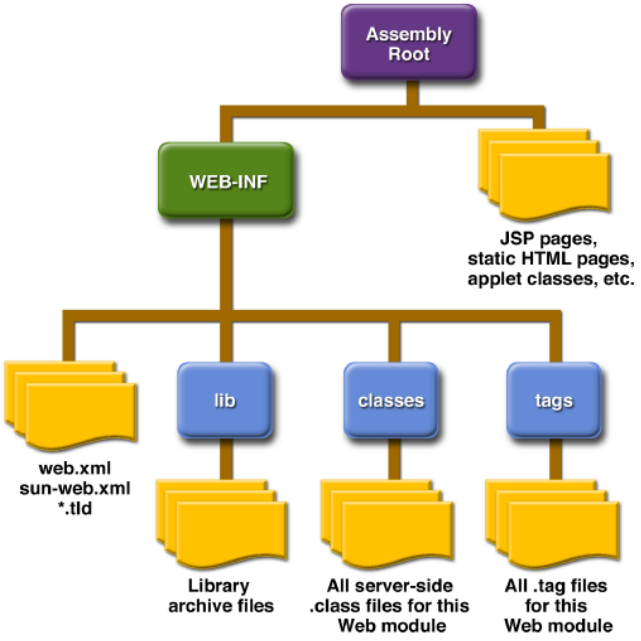

Figura 2. Estructura de una aplicación web java.

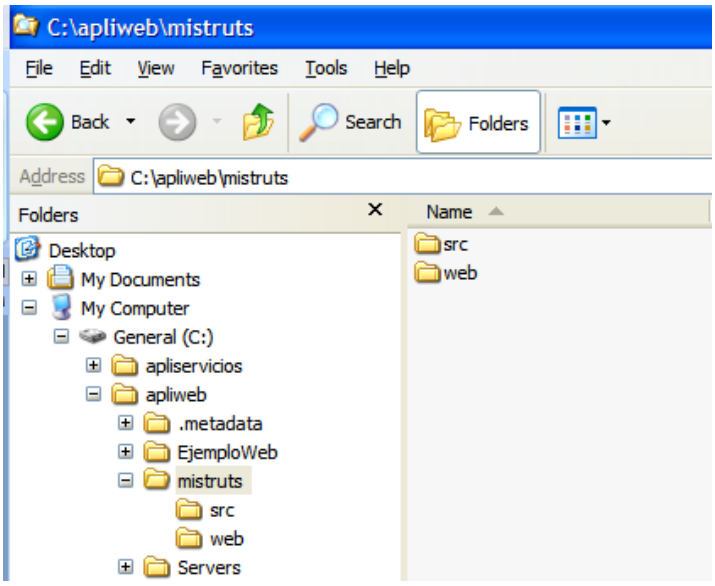

Figura 3. Estructura de directorios del proyecto.

- La estructura completa con el módulo web definido por la figura 2 quedaría tal como se muestra en la figura 4. El contenido del archivo web.xml no interesa por el momento puesto que el framework Struts lo cambia completamente a lo que ya se conoce. El archivo tradicional del descriptor de despliegue tiene un contenido similar a este:

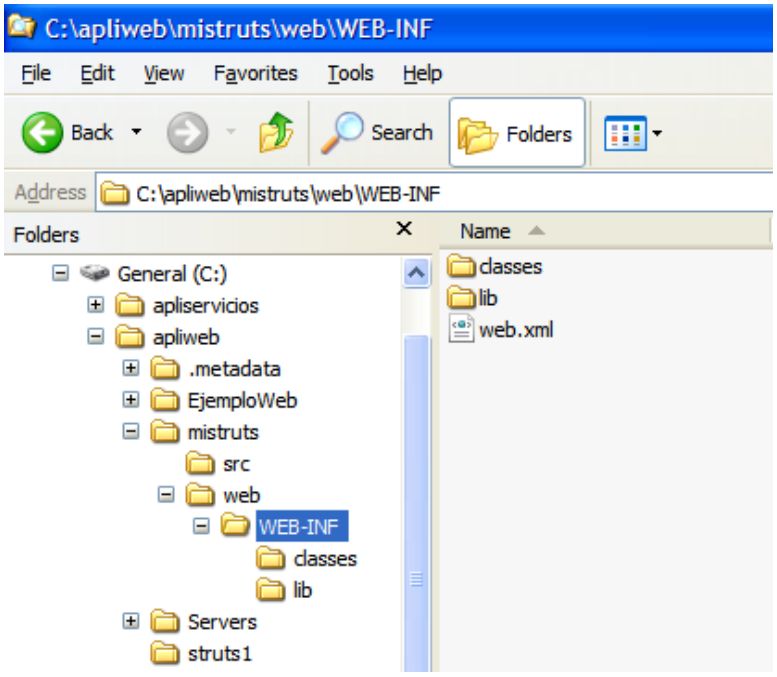

Figura 4. Estructura final del proyecto.

Archivo: **web.xml**

```
<?xml version="1.0" ?>
<web-app   version="2.5"   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-app_2_5.xsd">
  <display-name>apliservicios</display-name>
  <servlet>
    <servlet-name>ejemplo</servlet-name>
    <servlet-class>Ejemplo</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ejemplo</servlet-name>
    <url-pattern>/ejemplo</url-pattern>
  </servlet-mapping>
</web-app>
```

El modelo tradicional

- En una aplicación tradicional, las interfaces se construyen usando JSP y la lógica de la aplicación se implementa usando Servlets y clases java auxiliares para el acceso a datos. Cuando una petición de un cliente en Internet llega al servidor, el contenedor web recibe esa petición y la procesa, es decir, detecta cual es el recurso al que el usuario desea acceder (una JSP o un servlet) y lo busca en su interior. Para poder hacer esto, el archivo XML de configuración (web.xml) tiene configurado cada uno de los elementos que hacen parte de la aplicación. Realmente, el único elemento de configuración obligatoria es el servlet.
- Bajo este modelo tradicional, el contenedor se encarga de entregar a cada cliente, el recurso que solicito. ¿Cómo funciona Struts? Veamos el modelo Struts un poco en detalle para comprender lo que se hará en la práctica.

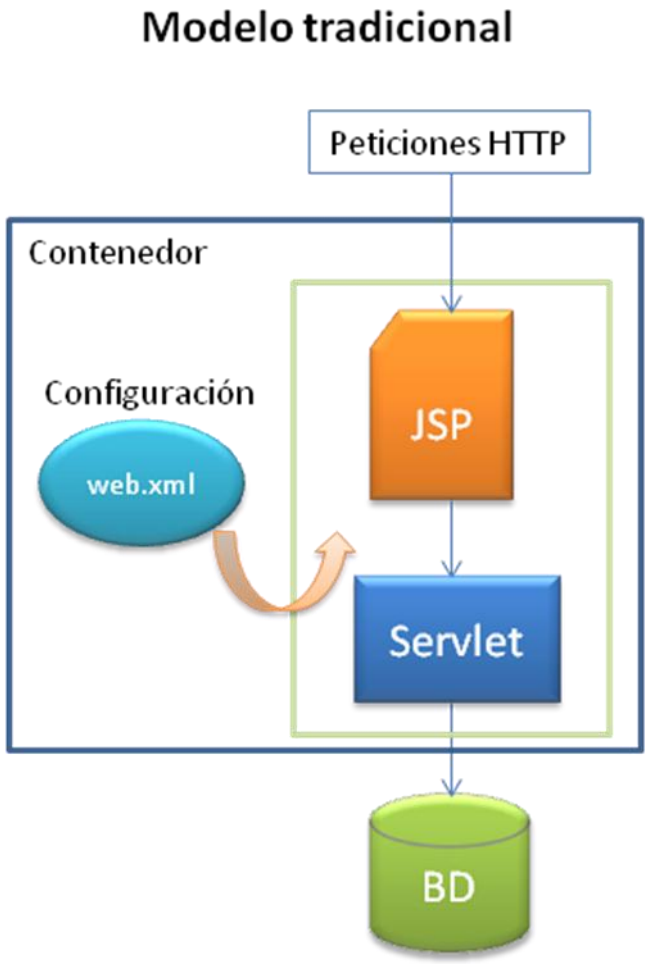

Figura 5. Modelo tradicional de aplicación web Java.

El modelo Struts

- En el modelo struts se realizan muchos cambios con el objetivo de facilitar las tareas del desarrollador y que este pueda concentrar su atención en resolver el problema de negocio y no malgastar esfuerzo innecesario en otras tareas más comunes que no tienen nada que ver con el problema real.
- En resumen, el modelo Struts funciona de la siguiente forma (ver figura 6): Cuando una petición de un cliente en Internet llega al servidor, el contenedor realiza la búsqueda del recurso solicitado pero, la aplicación ya no responde directamente.
- Ahora existe un intermediario conocido como **ActionServlet** que actúa como un enrutador de red (router) o controlador central, es bastante similar al patrón Fachada o al patrón Service Delegate. Toda petición pasa por este componente y este decide a quien entregarla para continuar con la tarea.
- La configuración de la aplicación web a través del archivo web.xml se hace más

simple puesto que ahora le deja el control al ActionServlet y por lo tanto, el único elemento configurado en este archivo será el ActionServlet.

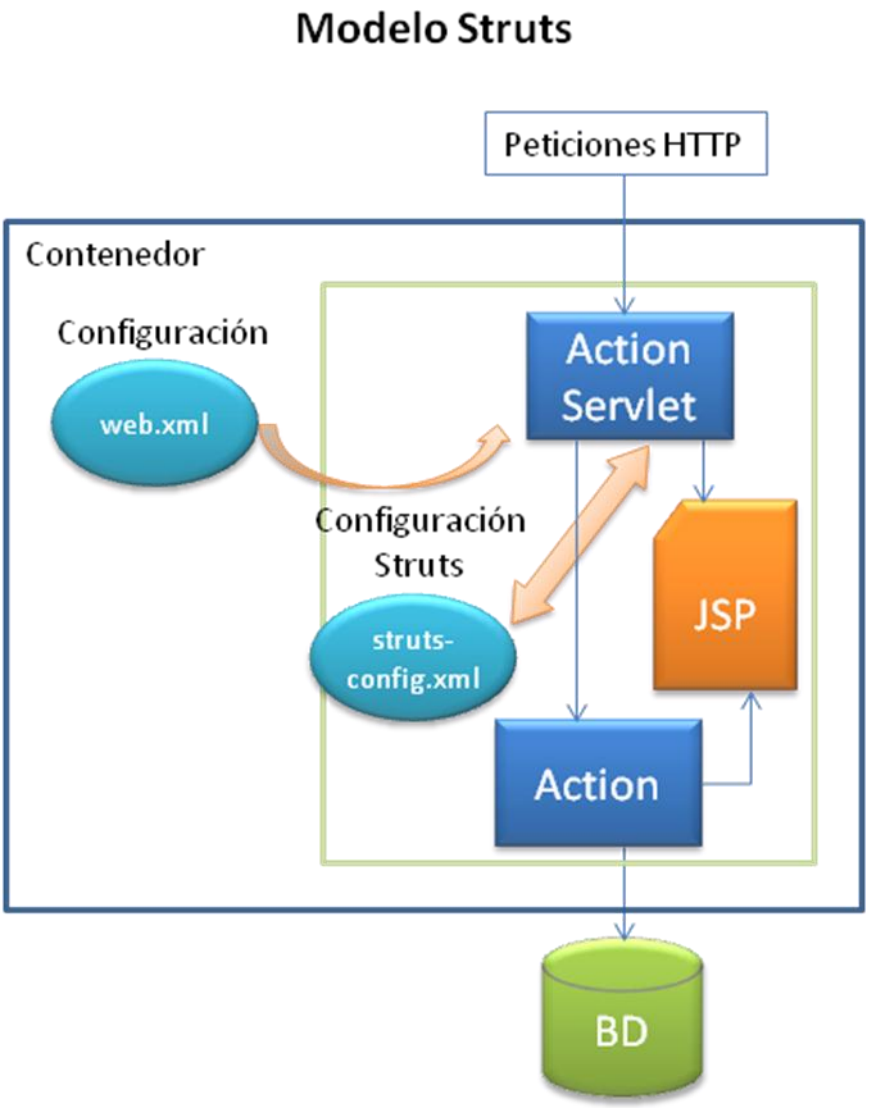

Figura 6. Modelo Struts.

- Este controlador central se encarga de analizar la solicitud, seguir el comportamiento configurado que se le ha programado en un archivo XML y llamar al Action correspondiente pasándole los parámetros enviados. Ahora los servlets serán conocidos como **Actions**. Es decir, el control sobre el flujo o la navegación dentro de la aplicación corre por cuenta del mismo Struts a través de un archivo XML denominado **struts-config.xml**. En este archivo se describe el comportamiento de la aplicación y los diferentes componentes definidos por el framework Struts.

- Desde aquí, la aplicación es capaz de decidir cuál es el recurso que se está solicitando o qué se le debe entregar al cliente en Internet. Según el resultado que retorne el Action (Servlet), el controlador invocará la generación de una interfaz de usuario a una o más JSPs, las cuales podrán consultar los objetos (clases java auxiliares) del Modelo con el fin de realizar su tarea.
- La lógica de la aplicación reside, generalmente en las clases auxiliares como los Java Beans de acceso a las bases de datos u otras clases java que el desarrollador quiera implementar. Los Actions (servlets) se encargan de ejecutar estas clases auxiliares y tomar la decisión del camino a seguir dentro de la lógica de navegación de la aplicación. Así por ejemplo, en un proceso de login, el Action usa una clase auxiliar para consultar la información del cliente y realizar la respectiva validación, y dependiendo del resultado, selecciona el camino a seguir, es decir, la interfaz de usuario que debe presentar al cliente, una página de mensajes o la interfaz de bienvenida de la aplicación, por ejemplo. Todo esto definido dentro del archivo XML de Struts.
- Dentro del archivo de configuración de Struts existe una sección para el mapeo de todas las acciones u objetos Action, en esta sección denominada **action-mappings**, la configuración de una Action (o servlet de struts) tiene este aspecto:

```
<action-mappings>
<action
      path="/Login"
      type="mistruts.LoginAction"
      name="loginForm"
      scope="request"
      validate="true"
      input="/pages/Login.jsp">
      <forward
            name="success"
            path="/pages/Welcome.jsp"/>
</action>
</action-mappings>
```

- Aquí están las instrucciones para invocar y controlar la ejecución de este servlet o Action, el atributo **path** corresponde al nombre usado en la URL para invocar la Action o Servlet. Por ejemplo, según este archivo de ejemplo:

http://localhost:8080/mistruts/Login. Debe empezar con el carácter "/".

- En el atributo **type** se encuentra el nombre completo de la clase Java que implementa el Action. Para este ejemplo: mistruts.LoginAction, es decir, la clase java LoginAction dentro del paquete java mistruts.
- El atributo **name** contiene el nombre del objeto **Form Bean** que está asociado con esta Action. Aquí es importante introducir la definición de lo que es un FormBean. Se trata de un objeto java auxiliar usado en Struts para la transferencia y formateo de los datos enviados desde un formulario de datos (generalmente implementado con una página JSP) hacia la Action (servlet).

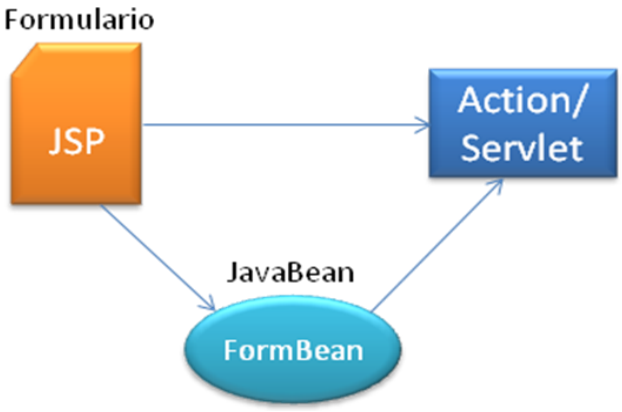

Figura 7. Definición de un FormBean.

- Cuando se crea una interfaz de usuario, se usan JSPs. En el caso particular de los formularios de datos, se envía también información. Esta información siempre viaja en formato de texto plano, sin embargo, java debe tratarlos en el formato o tipo de dato adecuado. El problema radica en el proceso de conversión de datos a los formatos apropiados. Struts ofrece esta facilidad a través de una clase auxiliar llamada **FormBean**. Esta clase está directamente vinculada con la interfaz de usuario (JSP) y con la **Action** que será invocada por este formulario, tal como se puede apreciar en la figura 7. El Action recibe la petición y crea una instancia del FormBean que se le ha configurado, de acuerdo con la JSP que le envía la petición, para eso, esta información se encuentra en la configuración de la aplicación en el atributo **name** de la etiqueta **action** dentro del archivo **struts-config.xml**.
- El atributo **scope** define el contexto en el cual la **Action** puede encontrar el FormBean. El **FormBean** recibe también el nombre de **ActionForm** desde el punto de vista de la **Action**. Son en realidad la misma cosa aunque se trate de

conceptos aparentemente diferentes. Es decir, el formulario JSP "envía" un FormBean y la Action "recibe" un ActionForm.

- El atributo **valídate** indica si se desea realizar validación de datos del lado del servidor a través del objeto ActionForm. Esto permite que se pueden, por ejemplo, mostrar mensajes de error en la interfaz del formulario.
- Por último, el atributo **input**, representa el path a otro recurso, por ejemplo una JSP, al cual se le debe devolver el control en el caso de que un error de validación ocurra. Solo es válido si se declara el atributo **name**.
- La etiqueta **forward** indica los posibles destinos después de finalizada la ejecución del Action. Se debe definir un nombre y la interfaz de usuario de destino.

Desarrollo de la aplicación Struts

- Partiendo de la estructura creada en la figura 4, se va a incluir el soporte de Struts a la aplicación web. Para que la aplicación web Java soporte el framework Struts se deben realizar los siguientes pasos:
- Primer paso: Copiar el directorio **lib** de la librería Struts que se descargo en el directorio **web**. En este documento se encuentra en la siguiente ubicación C:\apliweb\mistruts\web\WEB-INF. Ver la figura 8. No todos los archivos son necesarios, pero por facilidad, pasaremos el directorio completo.
- Segundo paso: modificar el archivo web.xml ubicado en el directorio C:\apliweb\mistruts\web\WEB-INF. En este archivo se debe indicar al servidor cual es el componente de Struts que será invocado. Es decir, aquí se configura el ServletAction, el componente principal de Struts que toma el control de la aplicación y actúa como enrutador. Para ello se elimina todo su contenido (es recomendable guardar una copia de seguridad de este archivo antes) y se debe escribir lo siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
      "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
      "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>Struts Blank Application</display-name>
```

```
  <!-- Standard Action Servlet Configuration -->
  <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>


  <!-- Standard Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>


  <!-- The Usual Welcome File List -->
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

Esto provoca que todas las solicitudes que lleguen sean atendidas por el **ActionServlet**. Indica también que el archivo de configuración para Struts será **struts-config.xml**, y que cualquier **action** de la aplicación será invocada con la terminación ***.do** o desde otro punto de vista, que cualquier invocación que termine en **.do** será capturada por Struts a través del ActionServlet.

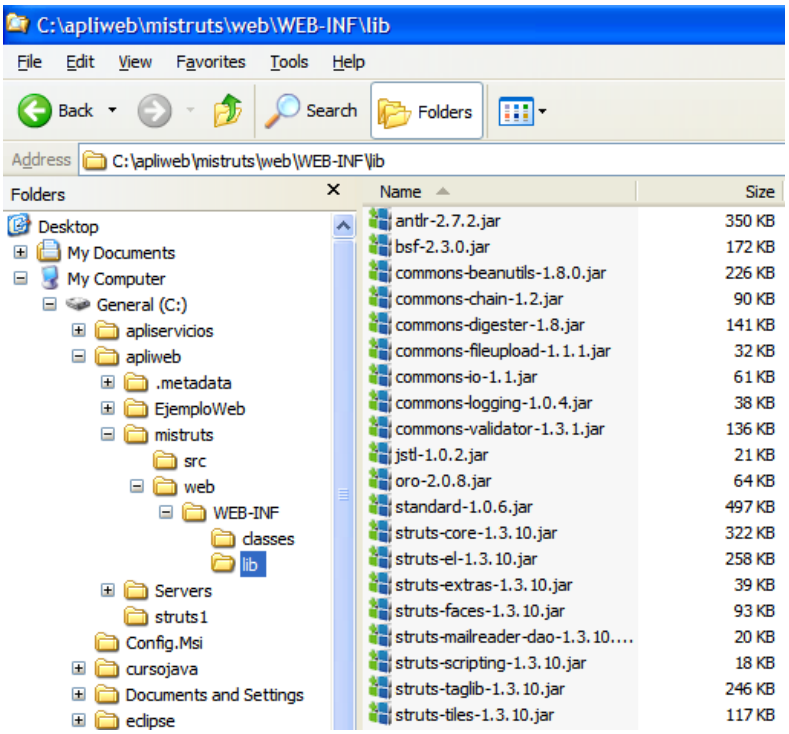

Figura 8. Librerías del framework Struts.

- Tercer paso: crear el archivo **struts-config.xml** en el directorio C:\apliweb\mistruts\web\WEB-INF. Un ejemplo del contenido de este archivo es el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
    <!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="loginForm"
                type="mistruts.form.LoginForm"/>
        </form-beans>
        <action-mappings>
            <action
                path="/Welcome"
                forward="/pages/login.jsp"/>
```

```
            <action
                path="/Login"
                type="mistruts.action.LoginAction"
                name="loginForm"
                scope="request"
                validate="true"
                input="/pages/login.jsp">
                <forward
                    name="success"
                    path="/pages/welcome.jsp"/>
                <forward
                    name="failure"
                    path="/pages/login.jsp"/>
            </action>
        </action-mappings>
        <message-resources
parameter="mistruts.ApplicationResources"/>
</struts-config>
```

- Este archivo representa el comportamiento de la aplicación web java que se muestra en la figura 9.

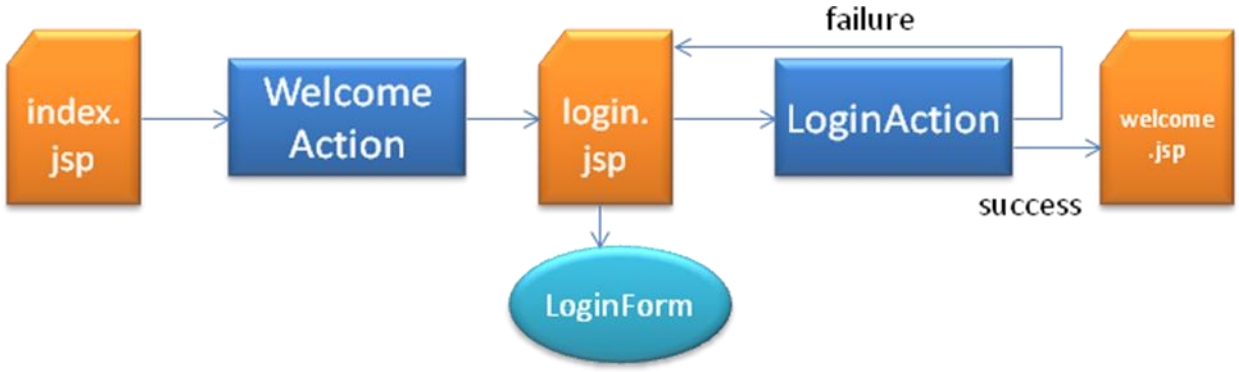

Figura 9. Aplicación web java de ejemplo.

- La acción **WelcomeAction** se encarga de cargar la página inicial de la aplicación únicamente.
- La acción **LoginAction** procesa los datos que llegan desde login.jsp y de acuerdo al resultado de la validación, decide redireccionar hacia **welcome.jsp** o nuevamente hacia **login.jsp**. Cada alternativa o redirección tiene su propio

nombre, **success** para ir a welcome.jsp y **failure** para ir a login.jsp, pero estos nombres solo son conocido por el Action llamado **LoginAction**.

- Cuarto paso: crear las paginas JSP, estas pueden hacer uso de las etiquetas especiales definidas en Struts escribiendo en la cabecera (inicio de la pagina) de cada JSP las líneas:

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

- Existen 4 tipos de etiquetas especiales para Struts. Las etiquetas Bean se usan para manejar objetos, especialmente para imprimir datos que llegan a una JSP a través clases java. Las etiquetas Html se usan para generar contenido dinamico e interactuar con java de una manera más simple. Las etiquetas logic sirven para sustituir la lógica escrita en lenguaje java por etiquetas similares a HTML.

Taller

- La tarea consiste en completar la aplicación para que permita realizar el proceso de validación de usuario usando Struts. No se necesita acceso a bases de datos, se puede hacer directamente sobre el código con un usuario predefinido.
- Recuerde que al momento de compilar, se deben incluir en el classpath las librerías de Struts que acaba de copiar.
- El código de los archivos básicos es:
- Archivo: **index.jsp**

```
<jsp:forward page="Login.do"/>
```

- La etiqueta **jsp:forward** se encarga de redireccionar o invocar a la acción **Login**. La acción welcome en su código java se encarga de redireccionar hacia la página principal. ¿Por qué hacer esta vuelta tan rara de invocaciones? Es una tarea normal y útil cuando se trabaja con **tiles** o interfaces graficas distribuidas en secciones. De lo contrario sobra.
- Los siguientes archivos se deben crear en el directorio

C:/apliweb/mistruts/web/pages/ dentro del directorio **web**. Archivo: **login.jsp**

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html"
prefix="html" %>
<html:html>
<head>
<title>Ingreso al sistema</title>
<html:base/>
</head>
<body bgcolor=”white”>
<html:form action="/LoginSubmit">
Usuario: <html:text property="login"/><html:errors
property="login"/><br/>
Clave: <html:password property="password"/><html:errors
property="password"/><br/>
<html:submit>Ingresar</html:submit>
<html:cancel/><html:reset/>
</html:form>
</body>
</html:html>
```

- Archivo: **welcome.jsp**

```
<%@ taglib uri="http://struts.apache.org/tags-bean"
prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html"
prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic"
prefix="logic" %>
<html:html>
<head>
<title><bean:message key="welcome.title"/></title>
<html:base/>
</head>
<body bgcolor="white">
<logic:notPresent name="org.apache.struts.action.MESSAGE"
```

```
scope="application">
  <font color="red">
    ERROR:  Application resources not loaded -- check servlet
container
    logs for error messages.
  </font>
</logic:notPresent>
<h3><bean:message key="welcome.heading"/></h3>
<p><bean:message key="welcome.message"/></p>
</body>
</html:html>
```

- En la página JSP anterior se usan las etiquetas **bean** para una función especial, obtener cadenas de texto paramatrizadas desde un archivo de recursos.. Struts cuenta con un archivo de recursos usado para internacionalización que se configura en el archivo struts-config.xml al final del mismo. El nombre por defecto de este archivo es **ApplicationResources**. Para ello se usa la etiqueta **message-resource**.
- En la página JSP, el parámetro **key** de la etiqueta **bean:message** señala cual cadena parametrizada cargar desde el archivo de recursos. Se pueden tener tantos archivos de recursos como idiomas quiera soportar, uno por cada idioma. Cada archivo debe finalizar con el código del idioma que soporta, por ejemplo, ApplicationResources_**de**.properties para el idioma alemán, ApplicationResources_**fr**.properties para francés, ApplicationResources_**en**.properties para ingles. Sin esa terminación será el idioma por defecto. De acuerdo con la configuración, este archivo debe estar ubicado en el directorio **C:/apliweb/mistruts/web/WEB-INF/classes/mistruts/**.

  Archivo: **ApplicationResources.properties**

  ```
  # -- welcome --
  welcome.title=Struts Blank Application
  welcome.heading=Welcome!
  welcome.message=Hello world!
  ```

- El código de las Actions se debe copiar en el directorio **C:/apliweb/mistruts/src/mistruts/action/**. El código de la acción WelcomeAction es:

Archivo: **WelcomeAction.java**

```
package mistruts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class WelcomeAction extends Action {

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

          return mapping.findForward("login");
     }
}
```

- El código de la acción LoginAction es

  Archivo: **LoginAction.java**

```
package mistruts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import mistruts.form.LoginForm;

public class LoginAction extends Action {
```

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

		LoginForm vuForm = (LoginForm) form;

		if (vuForm.getLogin().equals("javhur")) {
			if (vuForm.getPassword().equals("hola")) {
				return mapping.findForward("success");
			} else {
				return mapping.findForward("failure");
			}
		} else {
			return mapping.findForward("failure");
		}
	}
}
```

- Finalmente, se debe crear el objeto auxiliar FormBean o ActionForm, en el directorio **C:/apliweb/mistruts/src/mistruts/form/**.

  Archivo: **LoginForm.java**

```
package mistruts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class LoginForm extends ActionForm {
	private String login;
	private String password;

	public ActionErrors validate(ActionMapping mapping,
```

```
                    HttpServletRequest request) {

                    return null;
        }

        public      void      reset(ActionMapping      mapping,
    HttpServletRequest request) {

        }

        public String getLogin() {
              return login;
        }

        public void setLogin(String login) {
              this.login = login;
        }

        public String getPassword() {
              return password;
        }

        public void setPassword(String password) {
              this.password = password;
        }
}
```

- Compilar, empaquetar y publicar.
- Para compilar se puede crear un archivo con extension **bat**, por ejemplo, c.bat con los siguientes comandos, de acuerdo con la estructura de directorios que se tenga.

  Archivo: **c.bat**

```
set SRC_DIR=C:\apliweb\mistruts\src\mistruts

set DST_DIR=C:\apliweb\mistruts\web\WEB-INF\classes
```

```
set   LIB_DIR=C:\apliweb\mistruts\web\WEB-INF\lib\struts-core-1.3.10.jar

set      LIB2_DIR="C:\Program      Files\Apache      Software Foundation\Tomcat 6.0\lib\servlet-api.jar"

javac      -d      %DST_DIR%      -cp      %LIB2_DIR%;%LIB_DIR% %SRC_DIR%\action\*.java %SRC_DIR%\form\*.java
```

- Cada estudiante debe usar como datos de validacion su nombre de usuario y como password su codigo estudiantil.
- Probar en cualquier navegador.
- Fin de la práctica.
- Enviar la solucion por correo electronico a javhur@gmail.com
- Plazo: viernes, Diciembre 21 de 2012, hora: 11:59:59pm

Referencias

[1] Website del proyecto Struts. http://struts.apache.org