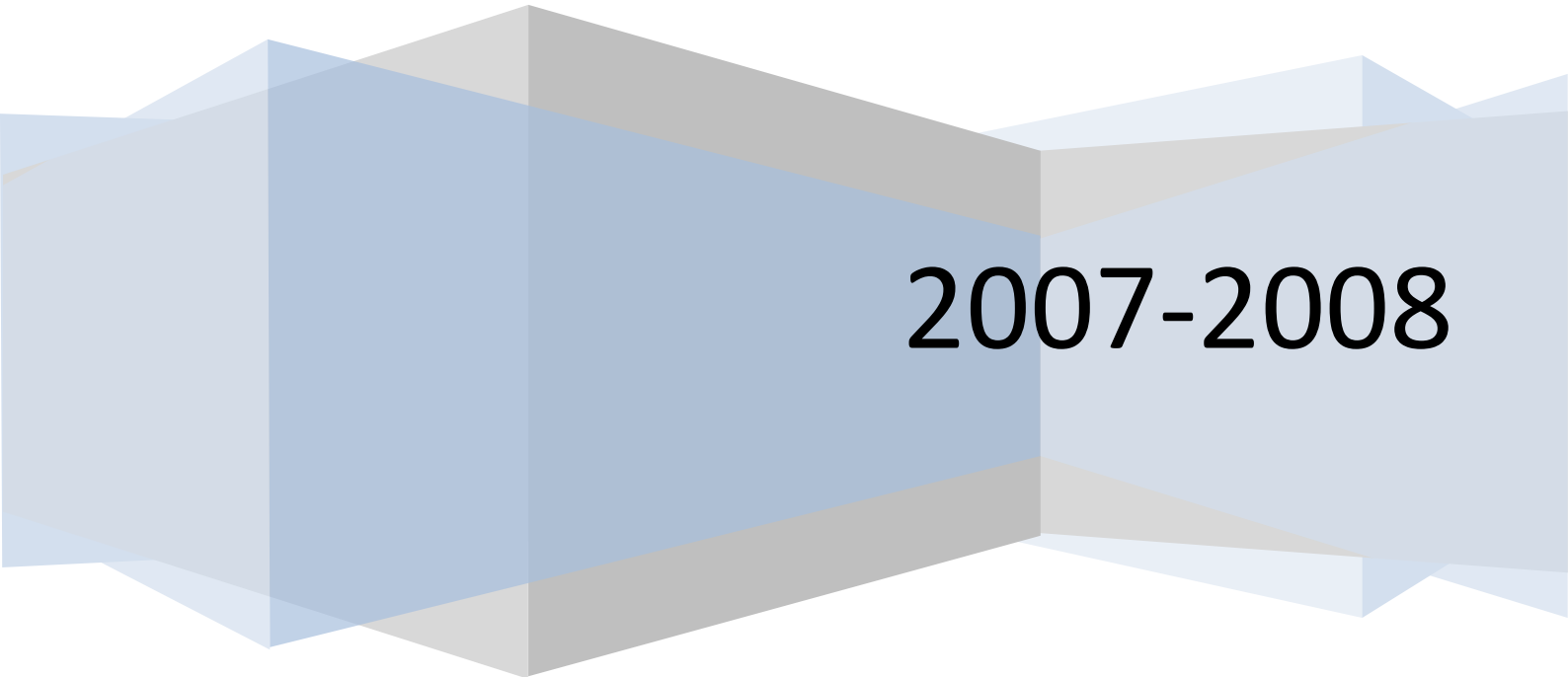


Universidad de Zaragoza

Apache Struts

Sistemas de Información

Fergus Reig Gracia 528055



2007-2008

Table of Contents

1.	Introducción	3
1.1.	¿Qué es Apache Struts?	3
1.2.	¿Qué es el Modelo Vista Controlador (MVC) ?	3
1.3.	¿Qué es Apache Software Foundation (ASF)?.....	3
1.4.	Otros.....	3
2.	Características principales.....	4
2.1.	Vista.....	4
2.2.	Controlador	6
2.3.	Modelo	7
3.	Comparativa con otras tecnologías.....	8
4.	Opinión del autor	10
5.	Tutorial	11
5.1.	web.xml	12
5.2.	plantillaUsoStruts.jsp	13
5.3.	NombreActionForm.java	14
5.4.	NombreAction.java	15
5.5.	struts-config.xml.....	16
5.6.	ConectarBD.java	17
5.7.	editarProfesorUsoStruts.jsp	18
5.8.	errorUsoStruts.jsp.....	19
5.9.	ApplicationResource.properties	19
6.	Instrucciones de instalación	20
7.	Software incluido	21
8.	Bibliografía	22
8.1.	Libros	22
8.2.	Páginas Web.....	22

Licencia

Copyright (c) 2008 Fergus Reig Gracia. Se permite la copia, distribución y/o modificación de este documento bajo los términos de la GNU Free Documentation License, Versión 1.1 o cualquier otra versión publicada por la Free Software Foundation.



1. Introducción

1.1. ¿Qué es Apache Struts?

Es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC sobre la plataforma J2EE (Java 2, Enterprise Edition), que generalmente se utiliza junto con bases de datos y páginas web escritas en JSP.

Struts se desarrollaba desde el año 2000 como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts permite reducir el tiempo de desarrollo. Su carácter de "software libre" y su compatibilidad con todas las plataformas en que Java Enterprise esté disponible, lo convierte en una herramienta altamente disponible.

1.2. ¿Qué es el Modelo Vista Controlador (MVC) ?

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio.

Descripción del patrón:

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

1.3. ¿Qué es Apache Software Foundation (ASF)?

Es una fundación no lucrativa creada en 1995 para dar soporte a proyectos de software, siendo el más famoso de ellos el servidor HTTP Apache.

Es una comunidad descentralizada de desarrolladores, cada uno de los cuales trabaja en sus propios proyectos de código abierto. Los proyectos Apache se caracterizan por un modelo de desarrollo basado en el consenso y la colaboración, y en una licencia de software abierta y pragmática.

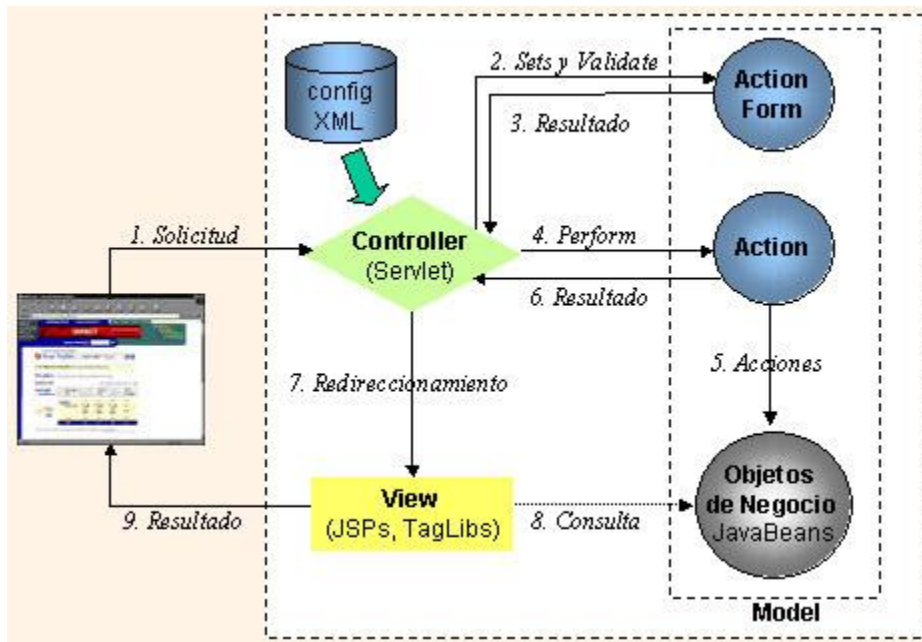
1.4. Otros

Suponemos que el usuario ya conoce al menos de forma vaga la plataforma J2EE y JavaServer Pages (JSP).



2. Características principales

Struts adapta el MVC de la siguiente manera:



2.1. Vista

La vista está generalmente compuesta por páginas JSP, Struts no modifica de forma importante el modo de utilización de estas páginas.

Notaremos el uso de Struts fundamentalmente en una disminución muy grande de la presencia de código Java que se mezcle con el HTML de la página. Este es en realidad uno de los objetivos principales de Struts.

Struts contiene 5 librerías de etiquetas para usar en los JSP, que si bien no son de uso obligatorio ayudan a sacar partido a algunas de las funcionalidades que nos da:

HTML: Sus etiquetas tienen relación una a una con las etiquetas <form> de HTML. Permiten conectar el componente Vista con el Controlador.

Bean: Sus etiquetas están relacionadas con la escritura de texto, evitan que los scriptlets muestren el contenido de los objetos almacenados en la petición o en la sesión y son necesarias para usar la función de internacionalización de Struts.

Logic: Sus etiquetas ayudan en el procesamiento condicional y la gestión de bucles, sustituyendo a los scriptlets.

Nested: Posee etiquetas para la visualización de propiedades anidadas dentro de un formulario u objeto.



Titles: Contiene etiquetas que permiten crear las distintas formas de ordenar elementos en un JSP.

ActionForm

Para el manejo de formas, Struts provee una clase llamada `ActionForm` que se debe extender para darle funcionalidad. Dentro de la subclase, por cada campo dentro de la forma, se debe declarar una variable instancia con el mismo nombre que se asignó al atributo `name` en el código HTML, para cada variable se deben escribir métodos `get` y `set`.

Adicionalmente se pueden sobrescribir los métodos `reset` y `validate` que tienen funcionalidades específicas. El método `reset` es invocado para reinicializar las variables a un valor dado por el programador, mientras el método `validate` se usa para validar que los datos no tengan errores de captura o valores nulos.

Internacionalización

Struts facilita la creación de aplicaciones multilenguaje mediante el uso combinado de las etiquetas de la librería `beans` y un archivo con extensión `properties` diferente para cada idioma.



2.2. Controlador

struts-config.xml

El desarrollador indica al framework las acciones, las formas y otros atributos mediante un archivo de configuración que por convención se llama **struts-config.xml**.

Mediante este fichero XML se especifican las interrelaciones entre acciones y páginas u otras acciones de forma centralizada en lugar de codificarlas en los propios programas o páginas. Este es uno de los grandes aciertos de Struts.

Action

Struts implementa el controlador mediante la clase **ActionServlet** del paquete **org.apache.struts.action**, es importante aclarar que esta clase es concreta, es el servlet principal de nuestra aplicación y no se necesita extender para poder usar el framework.

En este fichero recibiremos los datos de una página JSP mediante un **ActionForm**, llamaremos a las acciones del modelo y decidiremos cual es la siguiente página JSP que se debe lanzar.

La clase **actionServlet** es el motor principal de Struts, hace principalmente lo siguiente:

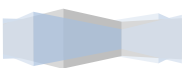
1. Recibe peticiones por parte de la vista, está puede ser un JSP.
2. Delega la petición a un objeto **RequestProcessor**, cuya clase se encuentra en el mismo paquete del **ActionServlet** y se encarga de seleccionar e invocar a la **Action** que corresponde a la petición. La clase **RequestProcessor** sigue para cada petición del usuario los siguientes pasos:
 - Identifica la acción que se quiere usar por la petición del usuario
 - Obtiene el nombre de la clase correspondiente a la acción.
 - Si es la primera petición a la clase, crea una instancia para su uso futuro
 - Llena una clase de tipo **ActionForm** asociada con esta petición, con los datos de la forma.
 - Se llama al método **execute** de la clase, pasando referencias al *bean* con los datos de la forma, al **HttpServletRequest** y al **HttpServletResponse** relacionado, así como un objeto **ActionMapping** que contiene información acerca de los recursos de donde proviene y hacia donde va el flujo de la navegación.
3. A su vez una **Action** manipula el estado y la lógica de nuestra aplicación y en lugar de mandar a la siguiente página, regresa un objeto **ActionForward** en el que se indica que recurso se encargará de la respuesta hacia el usuario.
4. El **RequestProcessor**, a través del objeto **ActionForward**, se encarga de mostrar o redireccionar a la siguiente página.



2.3. Modelo

En el modelo se lleva a cabo la lógica aplicativa del sistema Web que se está desarrollando. Struts no provee elementos que faciliten al desarrollador la implementación del modelo, ya que es responsabilidad del programador hacerlo.

Eso sí la estricta separación del código a la que nos obliga el uso de Struts hará que el código del modelo se encuentre centralizado, lo que lo hará a la larga más legible y fácil de actualizar. Esto ya puede considerarse por si solo una gran ventaja.



3. Comparativa con otras tecnologías

Hay muchas tecnologías similares a Struts, que intentan ayudarnos en los mismos procesos, algunas de ellas son:

Struts Shale: Una tecnología que se basa en JSF, también desarrollada por Apache y superior a Struts en algunos puntos pero que aun no cuenta con un número importante de usuarios.

ASP .NET: Es la herramienta para el desarrollo web de Microsoft, con lo bueno y malo que esto implica.

Apache Tapestry: Otro producto desarrollado por Apache, pero que no utiliza JSP como base.

WebObjects: Una de las primeras tecnologías de este tipo, actualmente en desuso.

Wotonomy: Reimplementación de código libre de WebObjects.

Y el competidor principal:

JavaServer Faces (JSF) es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL.

JSF es una especificación de Sun que cuenta actualmente con dos implementaciones la de referencia de la propia Sun y MyFaces de Apache.

Las diferencias básicas con Struts son:

Cierta información de configuración se ha trasladado de los archivos de configuración a los JSP: Desaparecen los archivos de configuración de Validation, formularios dinámicos, Titles, etc. Esta información se guardará en los propios JSP. Lo que si tendremos es un fichero principal de configuración que sustituirá al struts-config.xml, el fichero faces-config.xml.

El ciclo de vida del procesamiento de peticiones es más complejo: Como JSF utiliza componentes gráficos en el lado del servidor, procesa peticiones entrantes HTTP de forma diferente a Struts. El ciclo de vida de las peticiones JSF se divide en varias etapas, no como en Struts en el que solo tenemos dos fases: una para cuando el objeto ActionForm se valida, y la otra para cuando se invoca la función execute() de Action. Además, JSF permite a sus propias clases escuchar los eventos de cada clase, lo que supone una enorme ventaja sobre Struts.

JSF tiene como ventajas principales sobre Struts que:

Es una especificación de Sun, por lo que su supervivencia está asegurada.

La reutilización y extensibilidad del código fue una de las prioridades en el diseño de JSF.

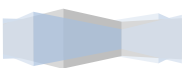
Es más sencillo crear código modular, lo que hace más fácil su mantenimiento y desarrollo posterior.



Ofrece formas muy estructuradas de extensión.

En aplicaciones Web muy complejas nos encontramos ante dificultades menores que con Struts.

Esto no quiere decir que JSF sea una opción mejor a Struts, siempre dependerá del caso particular, por ejemplo la mejor opción en caso de que trabajemos en una aplicación pequeña sería Struts ya que es más fácil de aprender y hace más sencillo desarrollar aplicaciones Web de pequeño tamaño.



4. Opinión del autor

Cuando comencé a preparar esta presentación general acerca de Apache Struts estaba iniciando mi PFC, que versa entorno a una aplicación Web que interactúa con una base de datos usando JSP para la realización de la página Web.

Apenas había hecho una página como plantilla que no me gustaba mucho, así que al empezar a ver en qué consistía Struts decidí rehacer dicha página con esta nueva tecnología que estaba conociendo.

Finalmente he decidido desechar la página que tenía inicialmente y tomar la que utiliza struts como referencia.

Esto es debido principalmente a que la utilización de un modelo como el MVC facilita muchísimo el desarrollo de aplicaciones en cuanto estas empiezan a ser de un tamaño medio y Struts nos facilita el uso de este modelo especialmente en lo referente a la interrelaciones entre las acciones y las página u otras acciones, permitiendo especificar estas interrelaciones mediante tablas XML en lugar de codificándolas en los programas o páginas.

Además es fácil de aprender, esto siempre es algo a tener en cuenta.



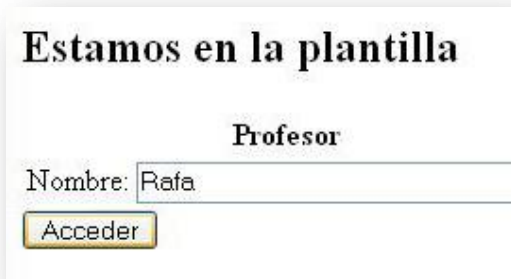
5. Tutorial

Este tutorial no explica la utilización de Struts dentro de ningún entorno de desarrollo concreto, por motivos didácticos se abstrae de esos detalles.

Desarrollaremos una pequeña aplicación con un sencillo formulario, explicando las instrucciones relacionadas con Struts sobre el propio código de ejemplo.

El resultado de nuestra aplicación será:

Una página inicial en la que deberemos introducir un nombre y pulsar sobre el botón aceptar:

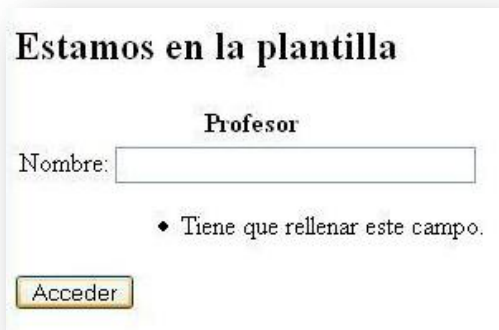


Estamos en la plantilla

Profesor

Nombre:

Si hemos dejado el campo en blanco se mostrará de nuevo la misma página, esta vez con un mensaje de error:



Estamos en la plantilla

Profesor

Nombre:

♦ Tiene que rellenar este campo.

Si todo ha ido bien se nos mostrará esta otra página, en la que mostramos el nombre, al que previamente hemos añadido, en la parte de la aplicación referente al modelo, la cadena "Struts":



Estamos en la página de respuesta

Profesor

Nombre: Rafa Struts



En caso de que se produzcan errores veremos la página:

No se ha podido completar la operación.

Algo ha ido muy mal.

5.1. web.xml

Lo primero que tenemos que hacer para poder usar Struts es modificar el archivo web.xml introduciendo dentro de la etiqueta <web-app> el código que sigue a continuación.

Web Pages/WEB-INF/ web.xml:

```
<!-- Definición del ActionServlet -->
<servlet>
  <!-- Nombre del servlet -->
  <servlet-name>action</servlet-name>

  <!-- Nombre completo de la clase ActionServlet-->
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <!-- Parámetro que contiene la ruta donde se encuentra nuestro archivo de configuración de struts -->
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
</servlet>

<!-- Todas las peticiones *.do son para el ActionServlet -->
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

<!-- Página de inicio de la aplicación -->
<welcome-file-list>
  <welcome-file>plantillaUsoStruts.jsp</welcome-file>
</welcome-file-list>
```



5.2. plantillaUsoStruts.jsp

El formulario será una página JSP, contiene una tabla con un solo campo de entrada y un botón para enviar los datos del formulario.

Web Pages/plantillaUsoStruts:

```
<%-- La página contiene código HTML --%>
<%@page contentType="text/html"%>
<%-- Tipo de codificación, para español es recomendable ISO-8859-1 --%>
<%@page pageEncoding="ISO-8859-1"%>
<%-- Página a la que llegaremos en caso de que se produzca un error en la página JSP --%>
<%@page errorPage="errorUsoStruts.jsp"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<%-- Librerías struts que utilizaremos--%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>

<html:html locale="true">
  <head>
    <title>Plantilla con Struts</title>
    <%-- Las rutas relativas tomarán como base la dirección de esta misma página --%>
    <html:base/>
  </head>
  <body>
    <h2>Estamos en la plantilla</h2>
    <%-- formulario que hace referencia a la acción recibirNombre.do --%>
    <html:form action="recibirNombre.do" focus="nombre">
      <table border="0">
        <thead>
          <tr> <th colspan="2">Profesor</th> </tr>
        </thead>
        <tbody>
          <tr valign="top">
            <td> Nombre: </td>
            <td>
              <%-- Campo de tipo texto de nombre "nombre" --%>
              <html:text property="nombre" size="32"/>
              <%-- Lugar donde se mostrará el error de nombre "blanco" --%>
              <html:errors property="blanco"/>
            </td>
          </tr>
        </tbody>
      </table>
      <%-- Botón que desencadena la acción --%>
      <html:submit> Acceder </html:submit>
    </html:form>
  </body>
</html:html>
```



5.3. NombreActionForm.java

Nuestro formulario tendrá asociado un ActionForm.

Source Packages/controlador/NombreActionForm.java

```
package controlador;

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

/** Se extiende la clase ActionForm de Struts */
public class NombreActionForm extends org.apache.struts.action.ActionForm {

    /** Esta variable sirve para guardar el nombre */
    private String nombre;

    /**
     * Sirve para obtener la variable nombre
     * @return variable String nombre
     */
    public String getNombre() {
        return nombre;
    }

    /**
     * Sirve para asignar un nombre a la variable
     * @param nombre cadena que se asignará
     */
    public void setNombre(String string) {
        nombre = string;
    }

    public NombreActionForm() {
        super();
    }

    /** Reinicializa las variable nombre a una cadena vacía. */
    public void reset(ActionMapping map, HttpServletRequest req){
        this.nombre = "";
    }

    /** Valida que el nombre no tengan errores de captura o valores nulos. */
    public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
        ActionErrors errors = new ActionErrors();

        /** En caso de que el nombre esté en blanco mostraremos el error "blanco". */
        if (getNombre() == null || getNombre().length() < 1) {
            errors.add("blanco", new ActionMessage("error.name.required"));
        }

        return errors;
    }
}
```



5.4. NombreAction.java

Nuestra clase Action recibirá los datos del formulario, llamará al Modelo para realizar las operaciones necesarias y decidirá cuál será la acción siguiente a realizar.

Source Packages/controlador/NombreAction.java:

```
package controlador;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionForward;

/** Importamos el paquete que contiene el modelo */
import modelo.*;

/** Se extiende la clase Action de Struts */
public class EditarAction extends org.apache.struts.action.Action {

    /** Sobre-escribimos el método execute */
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {

        // Recuperamos la forma
        NombreActionForm forma = (NombreActionForm) form;
        // Obtenemos el nombre introducido por el usuario
        String nombre = forma.getNombre();
        // Llamamos a un método del modelo
        ConectarBD modelo = new ConectarBD();
        nombre = modelo.otroNombre(nombre);
        // Agregamos al request el atributo nombre
        request.setAttribute("nombre", nombre);
        // Seleccionamos la siguiente pagina
        ActionForward fwd = mapping.findForward("mostrarProfesor");
        return fwd;
    }
}
```



5.5. struts-config.xml

En el fichero struts-config.xml definiremos gran parte de las interrelaciones que forman nuestra aplicación, esta es una de las ventajas principales de Struts.

Web Pages/WEB-INF/ struts-config.xml:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd" >

<!-- Esta etiqueta engloba la configuración de struts -->
<struts-config>

    <!-- Aquí definiremos nuestras ActionForms -->
    <form-beans>
        <!-- name = Nombre con el que haremos referencia a la clase -->
        <!-- type = Nombre completo de la clase -->
        <form-bean name="NombreActionForm"
            type="controlador.NombreActionForm"/>
    </form-beans>

    <!-- Aquí definiremos nuestras Actions -->
    <action-mappings>

        <!-- Definición de nuestra Action -->
        <!-- input = Página a la que redirige si el método validate del ActionForm devuelve error -->
        <!-- name = ActionForm que utiliza-->
        <!-- path = Nombre con el que llamamos a la Action sin ".do" (con una diagonal) -->
        <!-- type = Nombre completo de la clase -->
        <action
            input="/plantillaUsoStruts.jsp"
            name="NombreActionForm"
            path="/recibirNombre"
            scope="session"
            type="controlador.NombreAction">

            <!-- Esta etiqueta especifica las páginas que la clase Action puede llamar -->
            <!-- name = Nombre con el que llamaremos -->
            <!-- path = Nombre completo de la página -->
            <forward
                name="mostrarProfesor"
                path="/editarProfesorUsoStruts.jsp" />
            <forward
                name="errorProfesor"
                path="/errorUsoStruts.jsp" />
        </action>
        <action path="/Welcome" forward="/welcomeStruts.jsp"/>
    </action-mappings>

    <!-- Dirección en la que se encuentran los mensajes de error -->
```




```
<message-resources parameter="com/myapp/struts/ApplicationResource"/>

<!-- Definimos el tratamiento de excepciones -->
<global-exceptions>
  <!-- key = Nombre que daremos al error -->
  <!-- type = Nombre real de la excepción -->
  <!-- path = Página JSP a la que nos redirigiremos al producirse el error -->
  <exception key="errorBD"
             type="java.lang.IllegalArgumentException"
             path="/errorUsoStruts.jsp" />
</global-exceptions>
</struts-config>
```

5.6. ConectarBD.java

Esta clase se corresponderá con el “modelo” de nuestra aplicación, que podría por ejemplo interactuar con una base de datos.

No entraremos en detalle en esta parte, solo es importante volver a remarcar que el resto de la aplicación es completamente independiente.

Source Packages/modelo/ConectarBD.java:

```
package modelo;

import java.io.*;

/** No entra en la finalidad de este tutorial mostrar el funcionamiento interno del modelo. */
/** Lo normal sería que conectase con una base de datos y realizase diferentes operaciones. */
public class ConectarBD {

    /** Metodo constructor. */
    public ConectarBD () { }

    /** Procedimiento al que llamamos desde Action. */
    public String otroNombre (String nombre){
        return nombre + " Struts";
    }

}
```



5.7. editarProfesorUsoStruts.jsp

Esta es la página a la que se nos mandará desde la clase NombreAction.java, simplemente muestra por pantalla el nombre que se había leído.

En esta página podemos apreciar que no es obligatorio usar las etiquetas que pone Struts a nuestra disposición, podemos perfectamente usar las etiquetas normales de HTML.

Web Pages/editarUsoStruts.jsp:

```
<%-- La página contiene código HTML --%>
<%@page contentType="text/html"%>
<%-- Tipo de codificación, para español es recomendable ISO-8859-1 --%>
<%@page pageEncoding="ISO-8859-1"%>
<%-- Página a la que llegaremos en caso de que se produzca un error en la página JSP --%>
<%@page errorPage="errorUsoStruts.jsp"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <title>Respuesta con Struts</title>
  </head>
  <body>
    <h2>Estamos en la página de respuesta</h2>
    <table border="0">
      <thead>
        <tr>
          <th colspan="2">Profesor</th>
        </tr>
      </thead>
      <tbody>
        <tr valign="top">
          <td>
            Nombre:
          </td>
          <td>
            <%-- Mostramos el dato--%>
            <%= request.getAttribute("nombre")%>
          </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```



5.8. errorUsoStruts.jsp

En caso de que se produzca la excepción que definimos en struts-config.xml llegaremos a esta página.

También llegaremos a ella si se producen errores en cualquiera de las otras 2 páginas que usamos en el ejemplo, ya que dentro de ellas hemos definido esta como página de error, pero esto no es merced a Struts.

Web Pages/errorUsoStruts.jsp:

```
<!-- La página contiene código HTML -->
<%@page contentType="text/html"%>
<!-- Tipo de codificación, para español es recomendable ISO-8859-1 -->
<%@page pageEncoding="ISO-8859-1"%>
<!-- Identificamos esta página como usada para mostrar errores -->
<%@page isErrorPage="true"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!--Librerías struts que utilizaremos-->
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html" %>

<html:html locale="true">
  <head>
    <title>Mostrar error con Struts</title>
    <html:base/>
  </head>
  <body>
    <h2>No se ha podido completar la operación.</h2>
    <!-- Hacemos uso de la librería bean y el fichero properties para mostrar un mensaje configurable-->
    <p><bean:message key=" errors.general"/></p>
  </body>
</html:html>
```

5.9. ApplicationResource.properties

En nuestro caso este fichero solo contiene una línea, ya que solo lo utilizará plantillaUsoStruts.jsp para mostrar el error “blanco”.

Sin embargo este tipo de ficheros serán muy útiles en aplicaciones más grandes ayudando no solo a la gestión de los errores como en nuestro pequeño ejemplo, sino también a agilizar la tarea de mostrar nuestras páginas en diferentes idiomas.

Source Packages/com.myapp.struts/ApplicationResource.properties:

error.name.required = Tiene que rellenar este campo.

errors.general=Algo ha ido muy mal



6. Instrucciones de instalación

El código que forma parte del tutorial debe funcionar sobre cualquier servidor y entorno de programación compatible con JSP.

Sin embargo es importante saber que ha sido elaborado utilizando Netbeans.

Para probar el tutorial podemos:

- Instalar Netbeans.
- Copiar la carpeta Tutorial a nuestra máquina.
- Elegir en el menú “File” la opción “Open Project...”, y abrir el proyecto Tutorial que se encuentra en el CD.
- Para ejecutarlo solamente debemos pulsar F6 (Run Main Project)

Otra opción más didáctica es crear un proyecto nuevo:

- Instalar Netbeans.
- Elegir en el menú “File” la opción “New Project...”.
- En Choose Project elegir Web Application.
- En Name and Location elegir Apache Tomcat como servidor.
- En Frameworks elegir Struts.
- Realizar manualmente la introducción del código necesario*.
- Para ejecutarlo solamente debemos pulsar F6 (Run Main Project)

El DVD que acompaña el tutorial contiene también Eclipse, para ejecutar la aplicación propuesta en el tutorial en Eclipse tendremos bastantes más dificultades ya que tendremos que instalar Tomcat y la librería Struts manualmente.

Todo el material necesario está contenido en este DVD pero ninguna de estas tareas es inmediata. Podrá serle de utilidad en esta tarea algún otro tutorial, el siguiente es muy recomendable:

<http://www.laliluna.de/tutorial-struts-eclipse-espanol.html>

* No trivial



7. Software incluido

Todo el software incluido ha sido probado bajo Windows XP Service Pack 2, aunque la totalidad del software dispone de versiones para Linux y otros sistemas operativos.

Java:

Java EE 5 <http://java.sun.com/javaee/>

Entornos de desarrollo:

Eclipse 3.3.2 <http://www.eclipse.org/>

Netbeans IDE 6.0.1 <http://www.netbeans.org>

Servidor:

Apache Tomcat 6.0.16 <http://tomcat.apache.org/>

Librerías:

Struts 2.0.11.1 <http://struts.apache.org/>



8. Bibliografía

De estas fuentes se ha extraído gran parte de la información contenida en este tutorial, estas fuentes pueden ser utilizadas por el lector que así lo desee para profundizar en Struts.

8.1. Libros

Apache Struts

Arnaldo Doray

Anaya Multimedia - Apress, 2007. ISBN: 9788441522169

8.2. Páginas Web

Página oficial de Struts (en Inglés)

<http://struts.apache.org/>

Wikipedia

http://es.wikipedia.org/wiki/Apache_Struts

http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Manuales

http://www.myjavaserver.com/~aldosolis/tutorial_struts/struts_tutorial.html

<http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=miPrimeraWebStruts>

Tutoriales

<http://www.scribd.com/doc/97147/introduccion-al-framework-struts>

Y muchas páginas más que no son dignas de estar aquí consideradas.



