Contenido

Fundamentos de Struts

Struts es un framework que proporciona un conjunto de utilidades para facilitar y optimizar los desarrollos de aplicaciones Web con tecnología JavaEE, siguiendo el patrón MVC.
El empleo de Struts ofrece numerosos beneficios al programador, entre los que podríamos destacar:

- **Control declarativo de peticiones**. Controlar las acciones a realizar en función del tipo de petición que llega al controlador, este proceso es implementado por defecto por uno de los componentes que proporciona Struts, encargándose únicamente el programador de definir en un archivo de configuración XML los mapeos entre tipos de petición y acciones a ejecutar.

struts-config.xml

- **Utilización de direcciones virtuales**. Struts proporciona unos tipos de objetos que permiten referirse a estos recursos mediante direcciones virtuales asociadas a los mismos. La asociación entre una dirección virtual y su correspondiente URL o dirección real se realiza en un archivo de configuración.(forward strut-config.xml.)

- **Manipulación de datos con JavaBean**. Struts proporciona un sólido soporte para la manipulación de datos mediante JavaBeans, encargándose automáticamente el framework de la instanciación de los mismos, su rellenado con los datos procedentes del cliente y la recuperación y almacenamiento de los objetos en las variables de ámbito; todo esto sin necesidad de que el programador tenga que escribir una sola línea de código.

- **Juego de librerías de acciones JSP**. A fin de poder reducir al mínimo la cantidad de instrucciones de código Java dentro de una página JSP, Struts proporciona un amplio conjunto de librerías de acciones predefinidas con las que se pueden realizar la mayor parte de las operaciones que tienen lugar dentro de una página JSP.

La aportación de Struts al desarrollo de aplicaciones MVC se dirige básicamente a la construcción del Controlador y la Vista de una aplicación, dejando total libertad al programador en la implementación del Modelo y pueda así elegir la tecnología que crea más conveniente en cada caso.

Componentes

- **Archivos de configuración.**

web.xml:

descriptor de despliegue definido por una la especificación JavaEE.

struts-config.xml:

se registran y configuran los objetos de Struts.

```
<?xml version="1.0" encoding="UTF-8" ?>

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

<struts-config>
   <form-beans>
      <!--registro de FormBeans-->

   </form-beans>

   <global-exceptions>
                <!--definición de excepciones globales-->

   </global-exceptions>

   <global-forwards>
      <!--definición de objetos forward globales-->

   </global-forwards>

   <action-mappings>
      <!--asociación entre tipos de petición y objetos Action, además
                de objetos forward locales-->

   </action-mappings>

   <!--archivos de recursos-->
   <message-resources ... />
   :
</struts-config>
```

ApplicationResource.properties:

Archivo de texto plano para el almacenamiento de cadenas de texto en forma de parejas nombre=valor.

Cada vez que se requiera hacer uso de alguna de estas cadenas desde la aplicación, se hará utilizando la clave asociada a la misma. De esta manera, si el texto tiene que ser modificado, el cambio se realizará en el archivo de texto, no en el código.

Entre las utilidades de este archivo está la internacionalización de aplicaciones, pudiéndose definir tantos archivos de este tipo como idiomas se quiera mostrar en la aplicación,

```
<message-resources  parameter="paquete/ApplicationResource"/>
```

validator-rules.xml:

Contiene las reglas de validación predeterminadas por Struts, utilizadas para la validación automática de los datos de usuario.

validation.xml:

Archivo utilizado para la asignación de validadores a los campos de los formularios.

tiles-defs.xml:

los tiles o plantillas representan una de las características de Struts de cara a la creación de vistas.

▪ El API de Struts.

El API de Struts lo forman el conjunto de clases que el framework proporciona para estructurar las aplicaciones, siendo los paquetes más importante de todos org.apache.struts.action y org.apache.struts.actions.

ActionServlet

Un objeto ActionServlet es el punto de entrada de la aplicación, recibiendo todas las peticiones HTTP que llegan de la capa cliente.
Es básicamente de un servlet HTTP cuya clase hereda a HttpServlet.

El objeto ActionServlet cada vez que recibe una petición desde el cliente, extrae la última parte de la URL y la contrasta con la información contenida en el archivo de configuración struts-config.xml, a partir de la cual, se instancia el objeto JavaBean (ActionForm), lo rellena con los datos procedentes del formulario cliente y deposita la instancia en el contexto correspondiente, pasando a continuación el control de la petición al objeto Action encargado de procesarla.

Action

El Objeto Action es el responsables de procesar los distintos tipos de peticiones que llegan a la aplicación

El principal método con que cuenta esta clase es *execute(),* método que será invocado por ActionServlet al transferir la petición al objeto. Así por cada de petición que se vaya a controlar, el programador deberá definir una subclase de Action y sobrescribir el método execute(), incluyendo en él las instrucciones requeridas para el tratamiento de la petición, tales como llamadas a los métodos de la lógica de negocio implementada en el modelo o la transferencia de resultados a las vistas para su presentación.

ActionForm

El objeto ActionForm es un tipo especial de JavaBean que facilitan el transporte de datos. Es utilizado por *ActionServlet* para capturar los datos procedentes de un formulario y enviárselos al objeto *Action* correspondiente, sin utilizar *request.getParameter().*
Para ello, el programador deberá extender esta clase así como los correspondientes métodos set/get que den acceso a los datos.

ActionMapping.

Este objeto representa una asociación entre una petición y el objeto *Action* que la tiene que procesar. Contiene información sobre el path o tipo de URL que provoca la ejecución de la acción, así como de las posibles vistas que se pueden presentar al cliente tras su procesamiento.

ActionForward.

Este framework trabaja con direcciones virtuales en vez de reales. Las direcciones virtuales, más conocidas en Struts como forwards, son manejadas desde código a través de objetos *ActionForward*.
La clase *ActionForward* encapsula los detalles sobre la localización de un recurso, de modo que cada vez que se quiera transferir una petición desde código o redireccionar al usuario a una determinada página se hará utilizando objetos *ActionForward*.
Por ejemplo, la manera de indicar desde el método *execute*() de un objeto *Action* a *ActionServlet* que debe generar una determinada vista será devolviéndole el objeto *ActionForward* asociada a la misma.

▪ Librerías de acciones JSP.

HTML

Incluye acciones para la construcción de formularios HTML, incorporando parte de la funcionalidad encargada de la creación de beans de tipo ActionForm y el rellenado de los mismos con los valores de los controles. Además de este tipo de acciones, la librería html proporciona otros tags que facilitan la realización de tareas habituales en una página Web, como el enlace a otros recursos de la aplicación o la visualización de mensajes de error.

<html:html>. Genera la correspondiente etiqueta <html> con las características de localización propias del cliente.

<html:form>. Equivale a la etiqueta <form> y como en el caso de ésta sus principales atributos son *method* y *action*. El atributo *method* contiene el método de envío utilizado por la petición, siendo su valor por defecto "POST". En cuanto a *action*, contiene el path asociado al objeto Action que se encargará de tratar la petición, según la información indicada en struts-config.xml. Como se puede apreciar en el código de las páginas anteriores, la cadena indicada en *action* no incluye la extensión .do; una vez que la acción sea interpretada en el servidor y transformada en su equivalente etiqueta <form>, se establecerá en el atributo *action* de esta etiqueta la dirección relativa completa de la petición,

añadiendo a la izquierda del valor original el *context path* de la aplicación y a su derecha la extensión .do asociada a ActionServlet.

En el caso de login.jsp, y suponiendo que el directorio virtual de la aplicación Web es "/ejemplo", la acción:

<html:form action="/validar" method="post">

será transformada en:

<form action="/ejemplo/validar.do" method="post">

<html:text>. En vez de utilizar una única etiqueta para todos, Struts dispone de un tag diferente para la generación de cada uno de los controles gráficos de la interfaz, haciendo más intuitivo su uso. En el caso de <html:text>, genera una caja de texto de una única línea. Entre los atributos más importantes de esta acción se encuentran:

- **property**. Este atributo lo tienen todos los controles gráficos de la librería y representa el nombre de la propiedad del objeto ActionForm en la que se volcará el valor suministrado en el campo de texto. En caso de que la propiedad del bean disponga de un valor inicial, este será utilizado para inicializar el contenido del control.

- **maxlength**. Número máximo de caracteres que admite el control.

- **readonly**. Indica si el contenido del control es de sólo lectura (*true*) o de lectura y escritura (*false*).

- **onxxx**. Representa cada uno de los manejadores de eventos JavaScript que pueden ser capturados por el control, siendo *xxx* el nombre del evento.

<html:password>. Genera una caja de texto de tipo password, donde los caracteres introducidos se ocultan al usuario. Dispone de los mismos atributos que <html:text>.

<html:textarea>. Genera un control de texto multilínea.

<html:submit>. Equivale al control XHTML <input type="submit">. Al igual que éste, mediante su atributo *value* se establece el texto mostrado por el botón.

<html:reset>. Limpia todos los campos del formulario, dejando los valores por defecto establecidos. Equivale al contro XHTML <input type="reset">.

<html:select>. Genera una lista de selección de opciones. Además de *property*, este elemento dispone del atributo *size*, mediante el cual se indica si se generará una lista desplegable (size<=1) o una lista abierta (size>1).

<html:option>. Se emplea para la generación de las opciones en el interior de una lista de selección. En su atributo *value* se especifica el valor que será asignado a la propiedad del bean especificada en el atributo *property* de <html:select>, cuando la opción sea seleccionada.

<html:options>. Cuando las opciones de la lista se deben generar dinámicamente a partir del contenido de una colección, resulta más cómodo utilizar este elemento en vez del anterior. En su atributo *name* se indica el nombre del bean, existente en alguno de los ámbitos de la aplicación, que contiene la colección de datos a mostrar. Si la colección está contenida en

alguna de las propiedades del bean, se debería indicar en el atributo *property* el nombre de dicha propiedad.

<html:optionsCollection>. Es similar al anterior, aunque este tag se utiliza en aquellos casos en que la colección que contiene los elementos de la lista es de objetos de tipo JavaBean. Además de *name* y *property*, habrá que indicar en los atributos *label* y *value* los nombres de las propiedades del bean que representan el texto y valor de la opción, respectivamente. El siguiente bloque de código de ejemplo genera una lista con los datos de los objetos de tipo Libro contenidos en una colección llamada "books", utilizando las propiedades *titulo* e *isbn* del bean como texto y valor de cada opción:

```
<html:select>
        <html:optionsCollection name="books" label="titulo" value="isbn"/>
<html:select>
```

<html:checkbox>. Genera una casilla de verificación. Uno de sus principales atributos, *value*, indica el valor que se asignará a la propiedad del bean especificada en *property*, siempre y cuando la casilla se encuentre activada al hacer el submit del formulario.

<html:radio>. Genera un control de tipo radiobutton. Al igual que ocurre con checkbox, el atributo value contiene el valor que será asignado a la propiedad del bean especificada en *property*. Todos aquellos controles radiobutton con el mismo valor de atributo *property* serán excluyentes entre sí, siendo el contenido del atributo *value* de aquél que esté seleccionado el que se almacenará en la propiedad del bean al realizar el submit del formulario.

Todos los tags deberán aparecer siempre dentro del elemento <html:form>.

Bean.

Esta librería incluye acciones para el acceso a propiedades de un bean y parámetros de la petición desde una página JSP. También incluye acciones para la definición de nuevos beans y su almacenamiento en el contexto de la aplicación.

message
Permite mostrar en la página un mensaje de texto almacenado en un archivo externo, concretamente en ApplicationResource.properties. Este tag es muy utilizado en internacionalización de aplicaciones, permitiendo que los textosespecíficos de cada idioma puedan residir en archivos de recursos externos en vez de ser introducidos directamente en la página.

Los principales atributos de esta acción son:

- **key**. Clave asociada en ApplicationResource.properties a la cadena de caracteres. Struts localizará en el archivo de recursos el texto que tenga la clave especificada en este atributo y lo mostrará en la página.
- **name**. En vez de suministrar directamente en *key* el valor de la clave se puede extraer ésta de un objeto, en cuyo caso el nombre o identificador del objeto se indicará en este atributo.
- **property**. Nombre de la propiedad del objeto indicado en *name* que contiene la clave del mensaje.
- **scope**. Ámbito del objeto especificado en *name*.

Suponiendo que tenemos registrada la siguiente entrada en el archivo de recursos:

```
struts.message=Struts is easy
```
el bloque de acciones que se muestra a continuación almacena el nombre de la clave anterior en la propiedad “clave” de un objeto de la clase javabeans.Datos, mostrando a continuación el mensaje asociado a dicha clave:

```
<jsp:useBean id="info" class="javabeans.Datos">
<jsp:setProperty name="info" property="clave"
value="welcome.message"/>
</jsp:useBean>
<b>Mensaje</b>:
<h2>
<bean:message name="info" property="clave"/>
</h2>
```

Logic.

Los tags incluidos en esta librería realizan las operaciones que normalmente se llevan a cabo utilizando las estructuras lógicas de un lenguaje de programación, como el recorrido de una colección o la evaluación de condiciones.

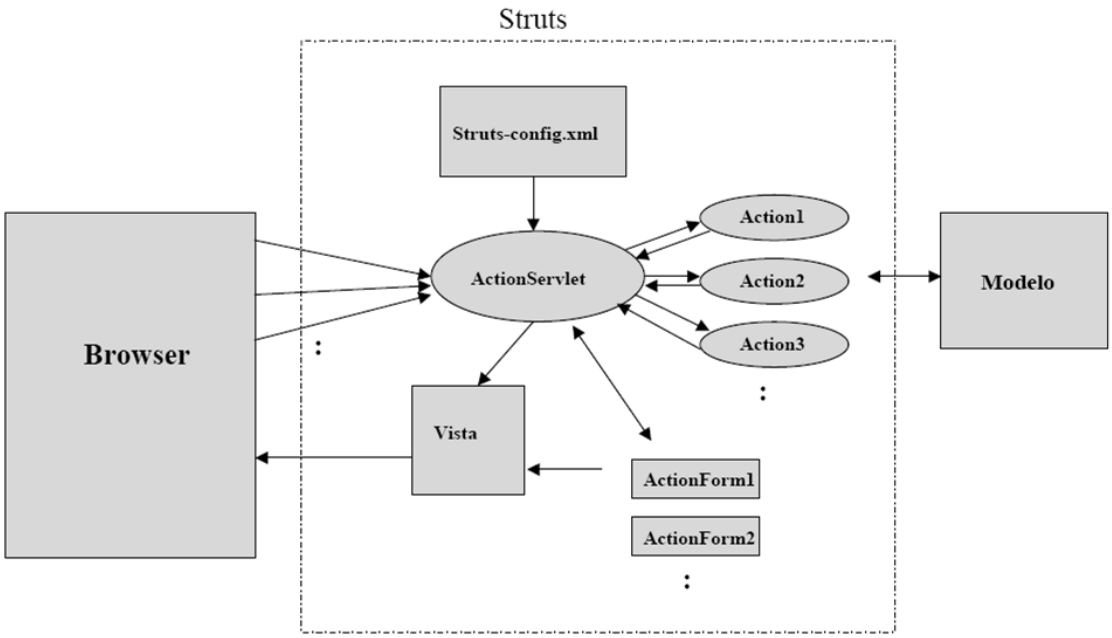
Nested.

Extiende parte de los tags de las librerías anteriores a fin de que puedan ser utilizados de forma anidada.

Tiles.

Esta librería incluye acciones para la definición de plantillas y su reutilización en diferentes páginas.

Funcionamiento



Análisis de la URL de la aplicación

El contenedor Web pasa la petición al objeto ActionServlet, éste, utiliza la última parte de la URL de la petición para determinar la acción a realizar.

Es habitual asignar como *url-pattern* del objeto ActionServlet el valor *.do, esto significa que cualquier URL procedente del cliente que termine en .do provocará que la petición sea capturada por este servlet.

Por ejemplo, suponiendo que el *context path* de una aplicación fuera "/ejercicio1" y estuviera desplegada en el servidor www.pruebas.com, las siguientes URL provocarían la ejecución de ActionServlet:

www.pruebas.com/ ejercicio1*/listado.do*

www.pruebas.com/ ejercicio1*/registro.do*

www.pruebas.com/ ejercicio1*/busqueda.do*

En este primer paso, ActionServlet extrae la parte de la URL que se encuentra entre el *context path* de la aplicación y la extensión .do, obteniendo en cada caso los valores "/listado", "/registro" y "/busqueda".

Determinación de la acción a realizar.

Utilizando el dato obtenido en el paso anterior, el objeto ActionServlet realiza una consulta en el archivo struts-config.xml para determinar las operaciones a realizar. Para cada tipo de acción el archivo de configuración define la subclase Action que debe ser instanciada, así como el objeto ActionForm asociado a la operación. Tras realizar la consulta en el archivo, ActionServlet lleva a cabo las siguientes acciones:

- Crea u obtiene la instancia del objeto ActionForm y lo rellena con los datos del formulario cliente.
- Crea una instancia del objeto Action correspondiente e invoca a su método *execute()*, pasándole como parámetro una referencia al objeto ActionForm.

Procesamiento de la petición

En el método *execute()* de la subclase Action correspondiente se codificarán las acciones para el procesamiento de la petición. Se debe procurar aislar toda la lógica de negocio en el Modelo, de manera que dentro de *execute()* únicamente se incluyan las instrucciones para interaccionar con los métodos de éste, además de aquellas otras encargadas del almacenamiento de resultados en variables de contexto para su utilización en las vistas.

Generación de la vista.

Como resultado de su ejecución, el método *execute()* devuelve a ActionServlet un objeto ActionForward que identifica al recurso utilizado para la generación de la respuesta. A partir de este objeto, ActionServlet extrae la dirección virtual encapsulada en el mismo y utiliza este valor para obtener del archivo struts-config.xml la dirección real de la página JSP correspondiente.

Funcionamento