

Laboratorio. Otros elementos de selección

Objetivo

Implementar una aplicación que permita realizar consultas sobre información de un usuario utilizando elementos de selección como “Radio Boxes” y “Check boxes” para establecer diferentes criterios de búsqueda.

Pasos a seguir

1. Desarrollar un nuevo proyecto que permita validar un usuario y despliegue un menú de opciones
2. Desarrollar una vista que permita elegir un criterio de búsqueda utilizando “Radio Boxes”
3. Desarrollar una vista que permita introducir un comentario y utilizar una “Check Box”
4. Desarrollar una vista que permita seleccionar uno o más criterios de búsqueda utilizando “Check boxes”
5. Desarrollar las clases necesarias para el llenado de información
6. Desarrollar las clases y vistas necesarias para desplegar la información

1. Desarrollar un nuevo proyecto

- Crear un nuevo proyecto llamado **TiposDeSeleccion**

Configurar el entorno de desarrollo

- Agregar las bibliotecas para manejo de Servlets y Struts
- Copiar las bibliotecas para el manejo de Struts a la carpeta **/WebContent/WEB-INF/lib**
- Crear los paquetes
 - **ts.struts.javabeans**
 - **ts.struts.modelo**
 - **ts.struts.servlets**
- Crear la carpeta paginas en **/WebContent**
- Copiar el archivo **struts-config.xml** de un proyecto en blanco al directorio **/WebContent/WEB-INF**
- Configurar el archivo **web.xml**
- Agregar y levantar el servidor Apache Tomcat

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <display-name>
        TiposDeSeleccion</display-name>
    <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>
        <init-param>
            <param-name>debug</param-name>
            <param-value>2</param-value>
        </init-param>
        <init-param>
            <param-name>detail</param-name>
            <param-value>2</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file></welcome-file>
    </welcome-file-list>
</web-app>
```

Programación de funcionalidad para Validar Usuario

JavaBean ValidarUsuarioForm

- En el paquete **ts.struts.javabeans** crear una clase llamada **ValidarUsuarioForm**

La clase ValidarUsuarioForm debe:

- Heredar de la clase **ActionForm**
- Tener los datos miembro login y password (ambos String)
- Tener sus métodos get y set
- Ser registrada como un JavaBean en el archivo **struts-config.xml**

ValidarUsuarioForm.java

```
package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class ValidarUsuarioForm extends ActionForm{

    private String login;
    private String password;

    public String getLogin() {
        return login;
    }
    public String getPassword() {
        return password;
    }
    public void setLogin(String login) {
        this.login = login;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

ActionServlet ValidarUsuarioAction

- En el paquete **ts.struts.servlets** crear una clase llamada **ValidarUsuarioAction**

La clase ValidarUsuarioAction debe:

- Heredar de la clase **Action**
- Su método `execute()` debe
 - Llama a la vista JSP del menú de opciones
 - Por el momento no se realiza ninguna validación real
- Ser registrada en el archivo **struts-config.xml**
 - Indicando que el JavaBean **ValidarUsuarioForm** estará activo durante toda la sesión

```
package ts.struts.servlets;

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 ts.struts.javabeans.*;

public class ValidarUsuarioAction extends Action{
```

```

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

        ValidarUsuarioForm vf = new ValidarUsuarioForm();
        vf=(ValidarUsuarioForm) form;

        return mapping.findForward("menuOpciones");

    }
}

```

Vista

La vista **login.jsp** creada en la carpeta **/WebContent/paginas** deberá:

- Solicitar el login y el password
- Asegurarse de limpiar el JavaBean **ValidarUsuarioForm** cada que se carga la página
- En su action form, indicar que se llamará al ActionServlet ValidarUsuarioAction

Apariencia



The screenshot shows a web browser window displaying a login form. The title of the page is "Introduce tus datos". Below the title, there are two input fields: "Login:" followed by a text box, and "Password:" followed by a text box. Below these fields, there are two buttons: "Aceptar" (Accept) and "Limpiar" (Clear). The form is simple and uses a standard web browser interface.

login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Página de validación</title>
</head>
<body>
<%session.removeAttribute("ValidarUsuarioForm"); %>
<h1>Introduce tus datos</h1>
<html:form action="/validarUsuario" method="POST">
<center>
Login: <html:text property="login"> </html:text><br><br>
Password: <html:password property="password"></html:password><br><br>
<br><br>
<html:submit property="submit" value="Aceptar"></html:submit>
<html:reset value="Limpiar"></html:reset>
</html:form>
</center>
</body>
</html>
```

Funcionalidad para manejar la opción

- Crear una clase en el paquete **ts.struts.javabeans** llamada **SeleccionarOpcionForm** que:
- Herede de la clase **ActionForm**
- Contenga un atributo llamado **opcionElegida** (String)
- Contenga sus métodos **get** y **set**
- Sea registrada en el archivo **struts-config.xml**

SeleccionarOpcionForm.java

```
package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class SeleccionarOpcionForm extends ActionForm{

    public String opcionElegida;

    public String getOpcionElegida() {
        return opcionElegida;
    }

    public void setOpcionElegida(String opcionElegida) {
        this.opcionElegida = opcionElegida;
    }

}
```

ActionServlet AdministrarOpcionAction

Servlet encargado de manejar la opción elegida en la vista, deberá :

- Heredar de la clase **DispatchAction**
- Contar con un método por cada una de las diferentes opciones que se puedan seleccionar
 - **todas()**
 - **unCriterioRadio()**
 - **unCriterioCaja()**
 - **varios()**
- Cada método llamará a una diferente vista
 - **todas()** **todoslosArticulos**
 - **unCriterioRadio()** **seleccionarRadio**
 - **unCriterioCaja()** **seleccionarCaja**
 - **varios()** **seleccionarCriterios**
- Ser registrada en el archivo **struts-config.xml** indicando que el nombre del método depende de un valor leído en el JSP
 - Registrar todos los **forwards** que estarán disponibles en este ActionServlet

AdministrarOpcionAction.java

```
package ts.struts.servlets;

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

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.actions.DispatchAction;

public class AdministrarOpcionAction extends DispatchAction{

    public ActionForward todas(ActionMapping mapping,
                               ActionForm form,
                               HttpServletRequest request,
                               HttpServletResponse response) throws Exception{

        return mapping.findForward("todosLosArticulos");
    }

    public ActionForward unCriterioRadio(ActionMapping mapping,
                                          ActionForm form,
                                          HttpServletRequest request,
                                          HttpServletResponse response) throws Exception{

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

```

    public ActionForward unCriterioCaja(ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception{

        return mapping.findForward("seleccionarCaja");
    }

    public ActionForward varios(ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception{

        return mapping.findForward("seleccionarCriterios");
    }
}

```

Vista

- Crear un JSP que despliegue las siguientes opciones
 - Mostrar todos los artículos
 - Elegir un criterio usando círculos de selección
 - Elegir un criterio usando cajas de selección
 - Elegir varios criterios de búsqueda
- Recordar que cada una de estas opciones llamará a un método diferente en AdministrarOpcionAction.

opciones.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Selección de búsqueda</title>
</head>
<body>
<html:form action="/administrarOpcion" method="POST">
<h1>Selecciona la búsqueda que quieras realizar</h1>
<center>
<html:select property="opcionElegida">
<html:option value="todas">Mostrar todos los artículos</html:option>
<html:option value="unCriterioRadio">Elegir un criterio usando
    círculos de selección</html:option>
<html:option value="unCriterioCaja">Elegir un criterio usando cajas de
    selección</html:option>
<html:option value="varios">Elegir varios criterios de
    búsqueda</html:option>

```

```

</html:select>
<br><br>
<html:submit>Aceptar</html:submit>
</center>
</html:form>
</body>
</html>

```

Desarrollar las vistas que serán llamadas por los diferentes forward dependiendo el método invocado, por el momento solo con título de página y encabezado indicando lo que se realiza.

- En cada página colocar una liga para regresar al menú de opciones y una para salir del sistema (regresar a la pantalla de login) Estos forward deberán ser registrados como globales en el archivo struts-config.xml

desplegarTodos.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Todos los artículos</title>
</head>
<body>
<h1>Tus artículos son</h1>
<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>
</body>
</html>

```

seleccionaCaja.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Selección por caja</title>
</head>
<body>
<h1>Selecciona tu criterio utilizando una caja</h1>
<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>
</body>
</html>

```


seleccionaRadio.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Selección por radio</title>
</head>
<body>
<h1>Elige un criterio utilizando Radio buttons</h1>

<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>

</body>
</html>
```

seleccionaCriterios.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Selección de múltiples criterios</title>
</head>
<body>
<h1>Selecciona los criterios de búsqueda que quieras</h1>

<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>

</body>
</html>
```

struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
  <data-sources />
  <form-beans>
    <form-bean name="ValidarUsuarioForm"
type="ts.struts.javabeans.ValidarUsuarioForm"></form-bean>
    <form-bean name="SeleccionarOpcionForm"
type="ts.struts.javabeans.SeleccionarOpcionForm"></form-bean>
  </form-beans>
  <global-exceptions />
  <global-forwards>
    <forward name="login" path="/paginas/login.jsp"></forward>
    <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
  </global-forwards>
  <action-mappings>
    <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
      <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
    </action>
    <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction"
parameter="opcionElegida">
      <forward name="todosLosArticulos"
path="/paginas/desplegarTodos.jsp"></forward>
      <forward name="seleccionarRadio"
path="/paginas/seleccionaRadio.jsp"></forward>
      <forward name="seleccionarCaja"
path="/paginas/seleccionaCaja.jsp"></forward>
      <forward name="seleccionarCriterios"
path="/paginas/seleccionaCriterios.jsp"></forward>
    </action>
  </action-mappings>
  <controller bufferSize="4096" debug="0" />
  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

Probar la aplicación (ejecutando el archivo login.jsp en el servidor)

A continuación se desarrollarán los distintos métodos de selección.

2. Selección utilizando Radio Buttons

Para cuestiones de ejemplo, se utilizará el JavaBean SeleccionarOpcionForm para contener todas las posibles opciones que se podrían seleccionar.

Para elegir con un Radio Button, se debe utilizar una variable tipo String, esta ya existe en el JavaBean (opcionElegida) por lo que no es necesario agregar ninguna otra.

Como este JavaBean ya está registrado en el archivo struts-config.xml tampoco es necesario registrarlo.

Creación de la vista

A sintaxis para utilizar un Radio Button es:

```
<html:radio property="atributo" value="valor">Texto</html:radio>
```

En donde:

- atributo corresponde al dato miembro del JavaBean que recibirá el dato leído
- valor representa la cadena que se recibirá cuando se elija ese RadioButton
- Texto representa el texto que aparece en la pantalla

Para poder utilizar los RadioButton es necesario agregar

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

Esta vista llamará al ActionServlet referenciado por el nombre “/administrarCriterios”, el código completo es el siguiente

seleccionaRadio.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Selección por radio</title>
</head>
<body>
<html:form action="/administrarCriterios" method="POST">
<h1>Elige un criterio utilizando Radio buttons</h1>
<center>
<html:radio property="opcionElegida"
value="buscarCD">Audio</html:radio><br><br>
<html:radio property="opcionElegida"
```

```

value="buscarPelicula">Películas</html:radio><br><br>
<html:radio property="opcionElegida" value="buscarVideoJuego">Video
juegos</html:radio><br><br>
<html:radio property="opcionElegida"
value="buscarTodos">Todos</html:radio><br><br>

<br><br>
<html:submit property="submit" value="Aceptar"></html:submit>
</center>
</html:form>
<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>

</body>
</html>

```

Lo siguiente es desarrollar el ActionServlet encargado de realizar las operaciones y manejar los datos enviados por la vista.

- Crear una clase llamada **AdministrarCriteriosAction** en el paquete **ts.struts.servlets**
- Se hará que cada opción llame a un método diferente, no solo al método execute
- Para esto, la clase debe extender a la clase **DispatchAction**
- Los nombres de los métodos deben coincidir con los escritos en los campos "value" de cada RadioButton
- Por el momento solo se imprime en pantalla la opción elegida

AdministrarCriteriosAction.java

```

package ts.struts.servlets;

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

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.actions.DispatchAction;

import ts.struts.javabeans.SeleccionarOpcionForm;

public class AdministrarCriteriosAction extends DispatchAction{

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

        SeleccionarOpcionForm of = new SeleccionarOpcionForm();
        of=(SeleccionarOpcionForm) form;
        System.out.println("CD");

        return null;

    }
}

```

```

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

        System.out.println("Pelicula");
        return null;
    }

    public ActionForward buscarVideoJuego(ActionMapping mapping,
                                           ActionForm form,
                                           HttpServletRequest request,
                                           HttpServletResponse response){
        System.out.println("Video juego");
        return null;
    }

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

        System.out.println("Todas");
        return null;
    }
}

```

Es necesario registrar el ActionServlet en el archivo struts-config.xml.

Como el método a llamar depende de un nombre, es necesario incluir el campo “parameter”

struts-config.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
    <data-sources />
    <form-beans>
        <form-bean name="ValidarUsuarioForm"
type="ts.struts.javabeans.ValidarUsuarioForm"></form-bean>
        <form-bean name="SeleccionarOpcionForm"
type="ts.struts.javabeans.SeleccionarOpcionForm"></form-bean>
    </form-beans>
    <global-exceptions />
    <global-forwards>
        <forward name="login" path="/paginas/login.jsp"></forward>
        <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
            <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>

```

```

</action>
<action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction"
parameter="opcionElegida">
    <forward name="todosLosArticulos"
path="/paginas/desplegarTodos.jsp"></forward>
    <forward name="seleccionarRadio"
path="/paginas/seleccionaRadio.jsp"></forward>
    <forward name="seleccionarCaja"
path="/paginas/seleccionaCaja.jsp"></forward>
    <forward name="seleccionarCriterios"
path="/paginas/seleccionaCriterios.jsp"></forward>
</action>
<action path="/administrarCriterios" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarCriteriosAction"
parameter="opcionElegida">
</action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

Se presenta un problema al ejecutar la aplicación y no seleccionar ninguno de los RadioButton, para evitar ese problema, es necesario colocarle un valor por defecto.

- Esto se realiza en el ActionServlet que manda llamar a la vista que contiene los radio buttons (AdministrarOpcionAction.java)
- Darle un valor a la variable que recibe el valor antes de llamar a la vista
- El método **unCriterioRadio** queda ahora:

```

public ActionForward unCriterioRadio(ActionMapping mapping,
                                   ActionForm form,
                                   HttpServletRequest request,
                                   HttpServletResponse response) throws Exception{
    SeleccionarOpcionForm of = new SeleccionarOpcionForm();
    of=(SeleccionarOpcionForm) form;
    of.setOpcionElegida("buscarTodos");
    return mapping.findForward("seleccionarRadio");
}

```

De ésta manera al cargarse la página, se tiene un valor por defecto.

3. Comentarios y caja de selección

El siguiente elemento es para manejar comentarios y una caja de selección, esta caja regresa un Booleano indicando si está activada o no, no regresa una cadena indicando su valor

Lo primero es incluir atributos en el ActionForm para que reciban estos valores

SeleccionOpcionForm.java

```
package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class SeleccionarOpcionForm extends ActionForm{

    private String opcionElegida;
    private boolean activada;
    private String comentario;

    public String getComentario() {
        return comentario;
    }

    public void setComentario(String comentario) {
        this.comentario = comentario;
    }

    public boolean isActivada() {
        return activada;
    }

    public void setActivada(boolean activada) {
        this.activada = activada;
    }

    public String getOpcionElegida() {
        return opcionElegida;
    }

    public void setOpcionElegida(String opcionElegida) {
        this.opcionElegida = opcionElegida;
    }

}
```

Lo siguiente es desarrollar la vista

Para el comentario se utiliza la instrucción

```
<html:textarea property="atributo" rows="10"
cols="30"></html:textarea>
```

En donde:

- atributo se refiere al nombre del atributo que recibirá el comentario

Para el checkbox se utiliza

```
<html:checkbox property="atributo">Texto </html:checkbox>
```

En donde:

- atributo se refiere al nombre del atributo que recibirá el valor de la caja (un Booleano)
- Texto es el texto a desplegar

Esta vista llamará a un ActionServlet referenciado por “/procesarComentario”

seleccionaCaja.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Selección por caja</title>
</head>
<body>
<html:form action="/procesarComentario" method="POST">
<h1>Manejo de comentarios y checkbox</h1>
Esta opción solamente regresa un booleano dependiendo si está
seleccionada o no
Por el momento, se utilizará para indicar si un comentario debe ser
publicado como anónimo
o no<br><br>
<center>
Introduce un comentario <br><br>
<html:textarea property="comentario" rows="10"
cols="30"></html:textarea><br><br>
<html:checkbox property="activada">Anónimo </html:checkbox>
<br><br>
<html:submit property="submit">Aceptar</html:submit>
<html:reset property="reset">Limpiar</html:reset>
</center>
</html:form>
<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>
</body>
</html>
```

Lo siguiente es desarrollar el ActionServlet

- Crear una clase llamada **AdministrarComentarioAction** en el paquete **ts.struts.servlets**
- Por el momento solo despliega en pantalla el comentario y el valor booleano

AdministrarComentarioAction.java

```
package ts.struts.servlets;

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 ts.struts.javabeans.SeleccionarOpcionForm;

public class AdministrarComentarioAction extends Action{

    public ActionForward execute(ActionMapping mapping,
                                ActionForm form,

    HttpServletRequest request,

    HttpServletResponse response)throws Exception{

        SeleccionarOpcionForm of = new SeleccionarOpcionForm();
        of = (SeleccionarOpcionForm) form;

        System.out.println(of.isActivada());
        System.out.println(of.getComentario());

        return null;
    }
}
```

Es necesario registrar este ActionServlet en el archivo struts-config.xml

struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
    <data-sources />
    <form-beans>
        <form-bean name="ValidarUsuarioForm"
type="ts.struts.javabeans.ValidarUsuarioForm"></form-bean>
        <form-bean name="SeleccionarOpcionForm"
type="ts.struts.javabeans.SeleccionarOpcionForm"></form-bean>
    </form-beans>
    <global-exceptions />
    <global-forwards>
        <forward name="login" path="/paginas/login.jsp"></forward>
        <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
```

```

scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
  <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
  </action>
  <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction"
parameter="opcionElegida">
    <forward name="todosLosArticulos"
path="/paginas/desplegarTodos.jsp"></forward>
    <forward name="seleccionarRadio"
path="/paginas/seleccionaRadio.jsp"></forward>
    <forward name="seleccionarCaja"
path="/paginas/seleccionaCaja.jsp"></forward>
    <forward name="seleccionarCriterios"
path="/paginas/seleccionaCriterios.jsp"></forward>
  </action>
  <action path="/administrarCriterios" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarCriteriosAction"
parameter="opcionElegida">
  </action>
  <action path="/procesarComentario" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarComentarioAction">
  </action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

4. Selección de múltiples criterios

Se desarrollará una página que permita seleccionar más de un criterio de búsqueda

Para recibir más de un criterio de búsqueda, se utiliza un arreglo de cadenas de caracteres, este debe registrarse en el ActionForm (SeleccionOpcionForm)

SeleccionOpcionForm.java

```
package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class SeleccionarOpcionForm extends ActionForm{

    private String opcionElegida;
    private boolean activada;
    private String comentario;
    private String[] opcionesSeleccionadas={};

    public String[] getOpcionesSeleccionadas() {
        return opcionesSeleccionadas;
    }

    public void setOpcionesSeleccionadas(String[]
opcionesSeleccionadas) {
        this.opcionesSeleccionadas = opcionesSeleccionadas;
    }

    public String getComentario() {
        return comentario;
    }

    public void setComentario(String comentario) {
        this.comentario = comentario;
    }

    public boolean isActivada() {
        return activada;
    }

    public void setActivada(boolean activada) {
        this.activada = activada;
    }

    public String getOpcionElegida() {
        return opcionElegida;
    }

    public void setOpcionElegida(String opcionElegida) {
        this.opcionElegida = opcionElegida;
    }

}
```

Para declarar UNA caja de selección que permanezca activa al seleccionar otra se utiliza la instrucción:

```
<html:multibox property="atributo" value="valor"></html:multibox>
```

En donde:

- atributo se refiere al atributo que recibirá las opciones elegidas (arreglo de cadenas de caracteres)
- valor es el valor de la cadena que identifica a esa opción
- El texto de la caja se coloca antes o después de la instrucción
- Esta vista llamará al ActionServlet referenciado por `"/administrarVariosCriterios"`

seleccionaCriterios.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Selección de múltiples criterios</title>
</head>
<body>

<html:form action="/administrarVariosCriterios" method="POST">
<h1>Selecciona los criterios de búsqueda que quieras</h1>
Películas<html:multibox property="opcionesSeleccionadas"
value="películas"></html:multibox><br><br>
Audio<html:multibox property="opcionesSeleccionadas"
value="cd"></html:multibox><br><br>
Video juegos<html:multibox property="opcionesSeleccionadas"
value="videoJuegos"></html:multibox><br><br>
<center>
<html:submit property="submit">Aceptar</html:submit>
<html:reset property="reset">Limpiar</html:reset>
</center>
</html:form>

<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>

</body>
</html>
```

El ActionServlet por el momento solo desplegará los elementos que se encuentren en el arreglo de cadenas

Crear una clase **AdministrarVariosCriteriosAction** en el paquete **ts.struts.servlets**

AdministrarVariosCriteriosAction.java

```
package ts.struts.servlets;

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 ts.struts.javabeans.SeleccionarOpcionForm;

public class AdministrarVariosCriteriosAction extends Action{

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

        SeleccionarOpcionForm of = new SeleccionarOpcionForm();
        of=(SeleccionarOpcionForm) form;

        String[] cadenas = of.getOpcionesSeleccionadas();

        for(int i=0;i<cadenas.length;i++)
            System.out.println(cadenas[i]);
        return null;

    }

}
```

Registrar el ActionServlet en el archivo struts-config.xml

struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
    <data-sources />
    <form-beans>
        <form-bean name="ValidarUsuarioForm"
type="ts.struts.javabeans.ValidarUsuarioForm"></form-bean>
        <form-bean name="SeleccionarOpcionForm"
type="ts.struts.javabeans.SeleccionarOpcionForm"></form-bean>
    </form-beans>
    <global-exceptions />
    <global-forwards>
        <forward name="login" path="/paginas/login.jsp"></forward>
        <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
```

```

    <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
  </action>
  <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction"
parameter="opcionElegida">
    <forward name="todosLosArticulos"
path="/paginas/desplegarTodos.jsp"></forward>
    <forward name="seleccionarRadio"
path="/paginas/seleccionaRadio.jsp"></forward>
    <forward name="seleccionarCaja"
path="/paginas/seleccionaCaja.jsp"></forward>
    <forward name="seleccionarCriterios"
path="/paginas/seleccionaCriterios.jsp"></forward>
  </action>
  <action path="/administrarCriterios" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarCriteriosAction"
parameter="opcionElegida">
  </action>
  <action path="/procesarComentario" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarComentarioAction">
  </action>
  <action path="/administrarVariosCriterios"
name="SeleccionarOpcionForm" scope="request"
type="ts.struts.servlets.AdministrarVariosCriteriosAction">
  </action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

Probar la aplicación (ejecutando el archivo login.jsp en el servidor)

5. Llenado de información

Se debe crear un ActionForm que contenga la información a desplegar por parte de los artículos

- Crear la clase **ListaDeArticulosForm** en el paquete **ts.struts.javabeans**
- Para simplificar el contenido de esta clase, solo contendrá un dato miembro de tipo **LinkedList** y sus respectivos métodos **get** y **set**
- Al ser un **javabeans**, debe heredar de la clase **ActionForm**

ListaDeArticulosForm.java

```
package ts.struts.javabeans;

import java.util.LinkedList;

import org.apache.struts.action.ActionForm;

public class ListaDeArticulosForm extends ActionForm{

    private LinkedList listaDeArticulos;

    public LinkedList getListaDeArticulos() {
        return listaDeArticulos;
    }

    public void setListaDeArticulos(LinkedList listaDeArticulos) {
        this.listaDeArticulos = listaDeArticulos;
    }

}
```

- También hay que crear la clase que contendrá la información que un artículo puede tener, en este caso:
 - Título
 - Género
 - Año
 - Tipo
- Esta clase llamada **Articulo** se creará en el paquete **ts.struts.modelo**
- No es necesario que herede de **ActionForm**
- Debe contener sus métodos **get** y **set**

Articulo.java

```
package ts.struts.modelo;

public class Articulo {

    private String titulo;
    private String genero;
    private String annio;
    private String tipo;

}
```

```

public Artículo (String ti, String t, String g, String a){

    tipo=ti;
    titulo=t;
    genero=g;
    annio=a;

}

public Artículo(){

}

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    public String getTitulo() {
        return titulo;
    }
    public String getGenero() {
        return genero;
    }
    public String getAnnio() {
        return annio;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    public void setGenero(String genero) {
        this.genero = genero;
    }
    public void setAnnio(String annio) {
        this.annio = annio;
    }
}

```

Llenado de las listas

La clase **RegresarArticulos** del paquete **ts.struts.modelo** es la encargada de llenar las listas (por el momento simulando el acceso a base de datos) de acuerdo al parámetro recibido.

RegresarArticulos.java

```
package ts.struts.modelo;

import java.util.LinkedList;

public class RegresarArticulos {

    public LinkedList todosLosArticulos(){

        LinkedList articulos=new LinkedList();

        Artículo articulo1 = new Artículo("Musica", "Bon
Jovi", "Rock", "2006");
        Artículo articulo2 = new Artículo("Musica", "Guns and
Roses", "Rock", "2000");
        Artículo articulo3 = new Artículo("DVD", "Viernes
13", "Terror", "2009");
        Artículo articulo4 = new Artículo("DVD", "Spider-Man
2", "Acción", "2004");
        Artículo articulo5 = new Artículo("Video juego", "God of
War", "Accion", "2003");
        Artículo articulo6 = new Artículo("Video juego", "FIFA
2009", "Deportivo", "2009");

        articulos.add(articulo1);
        articulos.add(articulo2);
        articulos.add(articulo3);
        articulos.add(articulo4);
        articulos.add(articulo5);
        articulos.add(articulo6);

        return articulos;
    }

    public LinkedList soloAudio(){

        LinkedList articulos=new LinkedList();

        System.out.println("Llenando Audio");

        Artículo articulo1 = new Artículo("Musica", "Bon
Jovi", "Rock", "2006");
        Artículo articulo2 = new Artículo("Musica", "Guns and
Roses", "Rock", "2000");

        articulos.add(articulo1);
        articulos.add(articulo2);

        return articulos;
    }
}
```

```

    public LinkedList soloDVD(){

        LinkedList articulos=new LinkedList();

        System.out.println("Llenando Video");

        Artículo articulo3 = new Artículo("DVD", "Viernes
13", "Terror", "2009");
        Artículo articulo4 = new Artículo("DVD", "Spider-Man
2", "Acción", "2004");

        articulos.add(articulo3);
        articulos.add(articulo4);

        return articulos;

    }

    public LinkedList soloVideoJuegos(){

        LinkedList articulos=new LinkedList();

        System.out.println("Llenando Video juegos");

        Artículo articulo5 = new Artículo("Video juego", "God of
War", "Accion", "2003");
        Artículo articulo6 = new Artículo("Video juego", "FIFA
2009", "Deportivo", "2009");

        articulos.add(articulo5);
        articulos.add(articulo6);

        return articulos;

    }

    public LinkedList variosCriterios(String [] criterios){

        LinkedList articulos=new LinkedList();
        LinkedList cd = new LinkedList();
        LinkedList dvd = new LinkedList();
        LinkedList vj = new LinkedList();

        for(int i=0;i<criterios.length;i++){

            if(criterios[i].compareTo("peliculas")==0)
                dvd=soloDVD();

            else if(criterios[i].compareTo("cd")==0)
                cd=soloAudio();

            else if(criterios[i].compareTo("videoJuegos")==0)
                vj=soloVideoJuegos();

        }

        articulos=unirListas(dvd,cd,vj);
        return articulos;

    }

```

```

    public LinkedList unirListas(LinkedList a, LinkedList b,
    LinkedList c){

        LinkedList unidades = new LinkedList();

        System.out.println(a.size()+" "+b.size()+" "+c.size());

        for(int i=0;i<a.size();i++)
            unidades.add(a.get(i));
        for(int i=0;i<b.size();i++)
            unidades.add(b.get(i));
        for(int i=0;i<c.size();i++)
            unidades.add(c.get(i));
        return unidades;

    }
}

```

Es necesario registrar en el archivo **struts-config.xml** la información del bean **ListaDeArticulosForm**.

struts-config.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.1//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
    <data-sources />
    <form-beans>
        <form-bean name="ValidarUsuarioForm"
type="ts.struts.javabeans.ValidarUsuarioForm"></form-bean>
        <form-bean name="SeleccionarOpcionForm"
type="ts.struts.javabeans.SeleccionarOpcionForm"></form-bean>
        <form-bean name="ListaDeArticulosForm"
type="ts.struts.javabeans.ListaDeArticulosForm"></form-bean>
    </form-beans>
    <global-exceptions />
    <global-forwards>
        <forward name="login" path="/paginas/login.jsp"></forward>
        <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
            <forward name="menuOpciones"
path="/paginas/menuOpciones.jsp"></forward>
        </action>
        <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction"
parameter="opcionElegida">
            <forward name="todosLosArticulos"
path="/paginas/desplegarTodos.jsp"></forward>
            <forward name="seleccionarRadio"
path="/paginas/seleccionaRadio.jsp"></forward>
            <forward name="seleccionarCaja"

```

```

path="/paginas/seleccionaCaja.jsp"></forward>
  <forward name="seleccionarCriterios"
path="/paginas/seleccionaCriterios.jsp"></forward>
  </action>
  <action path="/administrarCriterios" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarCriteriosAction"
parameter="opcionElegida">
  </action>
  <action path="/procesarComentario" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarComentarioAction">
  </action>
  <action path="/administrarVariosCriterios"
name="SeleccionarOpcionForm" scope="request"
type="ts.struts.servlets.AdministrarVariosCriteriosAction">
  </action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

6. Desplegando los artículos

Modificación de la vista

Se debe crear modificar el archivo JSP que despliega los artículos.

Esta vista nuevamente hará uso de beans e iteraciones para desplegar los elementos de una colección, en este caso una lista ligada, por esto es necesario incluir las instrucciones:

```

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

```

desplegarTodos.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Lista de artículos</title>
</head>
<body>
<h1>Tus artículos son</h1>
<center>
<table border="1">
<tr>
<td><b>Tipo</b></td>
<td><b>Nombre</b></td>

```

```

<td><b>Género</b></td>
<td><b>Año</b></td>
</tr>
<logic:iterate name="listaArticulos" property="listaDeArticulos"
id="articulo">
<tr>
<td><bean:write name="articulo" property="tipo"/></td>
<td><bean:write name="articulo" property="titulo"/></td>
<td><bean:write name="articulo" property="genero"/></td>
<td><bean:write name="articulo" property="año"/></td>
</tr>
</logic:iterate>
</table>
</center>
<html:link forward="menuOpciones">Regresar al menú de opciones
</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>
</body>
</html>

```

En particular, el bloque de instrucciones:

```

<logic:iterate name="listaArticulos" property="listaDeArticulos"
id="articulo">
<tr>
<td><bean:write name="articulo" property="tipo"/></td>
<td><bean:write name="articulo" property="titulo"/></td>
<td><bean:write name="articulo" property="genero"/></td>
<td><bean:write name="articulo" property="año"/></td>
</tr>
</logic:iterate>

```

Indica que “listaArticulos” es un atributo que se envía desde un ActionServlet, “listaDeArticulos” dice que ese atributo contiene un dato llamada “listaDeArticulos” que es una colección y con “articulo” se identifica a cada uno de los elementos de esa colección.

La primera opción manejada por el ActionServlet **AdministrarOpcionAction** llamará a una vista que despliegue todos los artículos.

Para poder realizar esto, primero hay que llamar a las clases que regresen una lista ligada con los artículos.

AdministrarOpcionAction.java

```

import org.apache.struts.action.ActionMapping;
import org.apache.struts.actions.DispatchAction;

import ts.struts.javabeans.ListaDeArticulosForm;
import ts.struts.javabeans.SeleccionarOpcionForm;
import ts.struts.modelo.Articulo;
import ts.struts.modelo.RegresarArticulos;

public class AdministrarOpcionAction extends DispatchAction{

```

```

    public ActionForward todas(ActionMapping mapping,
                               ActionForm form,
                               HttpServletRequest request,
                               HttpServletResponse response) throws Exception{

        ListaDeArticulosForm lf = new ListaDeArticulosForm();
        RegresarArticulos buscar = new RegresarArticulos();

        lf.setListaDeArticulos(buscar.todosLosArticulos());
        request.setAttribute("listaArticulos", lf);
        return mapping.findForward("todosLosArticulos");
    }

    public ActionForward unCriterioRadio(ActionMapping mapping,
                                           ActionForm form,
                                           HttpServletRequest request,
                                           HttpServletResponse response) throws Exception{

        SeleccionarOpcionForm of = new SeleccionarOpcionForm();
        of=(SeleccionarOpcionForm) form;
        of.setOpcionElegida("buscarTodos");

        return mapping.findForward("seleccionarRadio");
    }

    public ActionForward unCriterioCaja(ActionMapping mapping,
                                         ActionForm form,
                                         HttpServletRequest request,
                                         HttpServletResponse response) throws Exception{

        return mapping.findForward("seleccionarCaja");
    }

    public ActionForward varios(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response) throws Exception{

        return mapping.findForward("seleccionarCriterios");
    }
}

```

La instrucción

```
lf.setListaDeArticulos(buscar.todosLosArticulos());
```

Asigna la lista ligada que se regresó con los artículos al dato miembro **listaDeArticulos** de la clase **ListaDeArticulosForm** usando su método **set()**.

La instrucción

```
request.setAttribute("listaArticulos", lf);
```

Indica que un atributo llamado “listaArticulos” en la vista JSP recibirá un objeto tipo ListaDeArticulosForm (lf).

La instrucción

```
return mapping.findForward("todosLosArticulos");
```

Llama a la vista correspondiente.

Probar la aplicación (ejecutando el archivo login.jsp en el servidor)

Desplegando por un solo criterio

La clase **AdministrarCriteriosAction** es la responsable de desplegar en base a un solo criterio, completar esta clase es muy sencillo por que ya se tienen las clases que regresan listas ligadas de diferentes formas, solo se tienen que mandar a llamar, asignar la lista correspondiente al objeto **ListaDeArticulosForm** y llamar a la vista que los despliega (en este caso la misma vista que despliega todos).

AdministrarCriteriosAction.java

```
package ts.struts.servlets;

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

import ts.struts.javabeans.SeleccionarOpcionForm;

public class AdministrarCriteriosAction extends DispatchAction{

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

        ListaDeArticulosForm lf = new ListaDeArticulosForm();
        RegresarArticulos buscar = new RegresarArticulos();
        lf.setListaDeArticulos(buscar.soloAudio());
        request.setAttribute("listaArticulos", lf);
        return mapping.findForward("todosLosArticulos");
    }
}
```

```

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

        ListaDeArticulosForm lf = new ListaDeArticulosForm();
        RegresarArticulos buscar = new RegresarArticulos();

        lf.setListaDeArticulos(buscar.soloDVD());
        request.setAttribute("listaArticulos", lf);
        return mapping.findForward("todosLosArticulos");
    }

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

        ListaDeArticulosForm lf = new ListaDeArticulosForm();
        RegresarArticulos buscar = new RegresarArticulos();

        lf.setListaDeArticulos(buscar.soloVideoJuegos());
        request.setAttribute("listaArticulos", lf);
        return mapping.findForward("todosLosArticulos");
    }

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

        ListaDeArticulosForm lf = new ListaDeArticulosForm();
        RegresarArticulos buscar = new RegresarArticulos();

        lf.setListaDeArticulos(buscar.todosLosArticulos());
        request.setAttribute("listaArticulos", lf);
        return mapping.findForward("todosLosArticulos");
    }
}

```

Solo es necesario agregar en las acciones del ActionServlet correspondiente el forward hacia la vista referida por "todosLosArticulos" en el archivo **struts-config.xml**. Por lo que la información correspondiente al ActionServlet **AdministrarCriteriosAction** queda de la siguiente manera:

```

<action path="/administrarCriterios" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarCriteriosAction"
parameter="opcionElegida">
    <forward name="todosLosArticulos"
path="/paginas/desplegarTodos.jsp"></forward>
</action>

```

Probar la aplicación (ejecutando el archivo login.jsp en el servidor)

Manejo del comentario

Por el momento solo se llamará nuevamente a la vista de manejo de opciones, dejando el manejo del comentario y la casilla de activación desplegado solo en pantalla.

AdministrarComentarioAction.java

```
package ts.struts.servlets;

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 ts.struts.javabeans.SeleccionarOpcionForm;

public class AdministrarComentarioAction extends Action{

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

        SeleccionarOpcionForm of = new SeleccionarOpcionForm();
        of = (SeleccionarOpcionForm) form;

        System.out.println(of.isActivada());
        System.out.println(of.getComentario());

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

Como el forward a la vista referida por “menuOpciones” ya está registrada en los forward globales, no es necesario registrarlo nuevamente.

Probar la aplicación (ejecutando el archivo login.jsp en el servidor)

Manejo de varios criterios

La clase **AdministrarVariosCriteriosAction** ya desplegaba los elementos de la cadena de caracteres que contenía las opciones seleccionadas por el usuario, la clase **RegresarArticulos** ya tiene un método que recibe esa cadena y regresa la lista ligada correspondiente, por lo que solamente hay que mandarla llamar, asignar la lista al dato miembro **listaDeArticulos** del **ListaDeArticulosForm** y enviarsela al atributo correspondiente (“listaArticulos”) en el JSP.

AdministrarVariosCriteriosAction.java

```
package ts.struts.servlets;

import java.util.LinkedList;

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 ts.struts.javabeans.ListaDeArticulosForm;
import ts.struts.javabeans.SeleccionarOpcionForm;
import ts.struts.modelo.Articulo;
import ts.struts.modelo.RegresarArticulos;

public class AdministrarVariosCriteriosAction extends Action{

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

        ListaDeArticulosForm lf = new ListaDeArticulosForm();
        RegresarArticulos buscar = new RegresarArticulos();
        SeleccionarOpcionForm of = new SeleccionarOpcionForm();

        of=(SeleccionarOpcionForm) form;
        String[] cadenas = of.getOpcionesSeleccionadas();
        lf.setListaDeArticulos(buscar.variosCriterios(cadenas));
        request.setAttribute("listaArticulos", lf);
        return mapping.findForward("todosLosArticulos");
    }
}
```

Solo es necesario agregar en las acciones del ActionServlet correspondiente el forward hacia la vista referida por **"todosLosArticulos"** en el archivo **struts-config.xml**. Por lo que la información correspondiente al ActionServlet AdministrarVariosCriteriosAction queda de la siguiente manera:

```
<action path="/administrarVariosCriterios" name="SeleccionarOpcionForm"
scope="request"
type="ts.struts.servlets.AdministrarVariosCriteriosAction">
<forward name="todosLosArticulos"
path="/paginas/desplegarTodos.jsp"></forward>
</action>
```

Probar la aplicación (ejecutando el archivo login.jsp en el servidor)