

# Laboratorio. Manejo de opciones y consultas sencillas

## Objetivos

- Implementar una aplicación que permita realizar consultas de información
- Hacer que un JavaBean pueda ser utilizado desde cualquier JSP o servlet
- Conocer el funcionamiento de las tags de manejo de opciones para elegir una actividad

## Pasos a seguir

1. Desarrollar una navegación sencilla entre ventanas en base a la opción elegida en un menú de opciones
2. Implementar clases para la consulta de los datos de un usuario
3. Implementar clases para la consulta de artículos pertenecientes a un usuario

### 1. Desarrollo de navegación sencilla

Se comenzará un nuevo proyecto, por lo que es necesario volver a configurar el ambiente de desarrollo

- Ejecutar eclipse
- Crear un nuevo proyecto “Dynamic Web Project” llamado **LaboratorioOpcionesStruts**
- Crear la carpeta **paginas** dentro de **/WebContent**
- Agregar al proyecto las bibliotecas necesarias para trabajar con Servlets y Struts
- Agregar el servidor Tomcat
- Dentro de la carpeta del código, crear los paquetes:
  - **ts.struts.javabeans**
  - **ts.struts.modelo**
  - **ts.struts.servlets**
- Copiar las bibliotecas de Struts a la carpeta **/WebContent/WEB-INF/lib** del proyecto
- Copiar el archivo **struts-config.xml** en blanco al directorio **/WebContent/WEB-INF/** del proyecto

### Configuración del archivo web.xml

Como se recordará, en este archivo se debe configurar el servlet ActionServlet que es utilizado en la mayoría de las aplicaciones junto con todos sus parámetros

## 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>
        LaboratorioOpcionesStruts</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>
```

Crear una clase **ValidarUsuarioForm** en el paquete **ts.struts.javabeans** que contendrá los datos a ser leídos desde la pantalla de validación, al tratarse de un JavaBean, es necesario que herede de la clase **ActionForm**

### 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;
    }
}

```

Desarrollar la clase **ValidarUsuarioAction** en el paquete **ts.struts.servlets** que servirá para realizar la validación y llamar a la pantalla que ofrecerá las opciones al usuario.

#### ValidarUsuarioAction.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.*;

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("opciones");
    }

}

```

Como se observa, esta clase realiza un forward a una página referenciada por el valor **opciones**, esto se deberá registrar en el archivo **struts-config.xml**

Desarrollar la vista del archivo **login.jsp** creando un nuevo archivo **login.jsp** en la carpeta **WebContent/paginas**

## 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:html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Validación de usuario</title>
</head>
<body>
<html:form action="/validarUsuario" method="POST">
<center>
<h1>Introduce tus datos</h1>
Usuario:
<html:text property="login" /></html:text>
<br><br>
Password:
<html:password property="password"/></html:password>
<br><br>
<html:submit>Aceptar</html:submit>
<html:reset>Limpiar</html:reset>
</center>
</html:form>
</body>
</html:html>
```

Hasta el momento es necesario registrar un JavaBean llamado **ValidarUsuarioForm** y las acciones (incluyendo el forward hacia opciones) realizado por el Action servlet **ValidarUsuarioAction**, esta información se agrega al archivo **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-beans>
    <global-exceptions />
    <global-forwards>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
            scope="request" type="ts.struts.servlets.ValidarUsuarioAction">
            <forward name="opciones" path="/paginas/opciones.jsp"></forward>
        </action>
    </action-mappings>
    <controller bufferSize="4096" debug="0" />
    <message-resources
        parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

## Desarrollo de la vista opciones.jsp encargada de desplegar un menú de opciones

Para utilizar las tags que proporcionan el manejo de opciones, es necesario incluir la siguiente línea

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

El manejo de opciones se realice de la siguiente manera:

```
<html:select property="opcionElegida">
<html:option value="valor_1">Opción_1</html:option>
</html:select>
```

En donde:

“valor\_1” representa la clave con la que se identificará la opción Opción\_1

Es importante notar que dentro de <html:select> se incluye un campo llamado **property** cuyo valor debe ser el atributo de un JavaBean, por lo que es necesario crear uno que reciba la clave de la opción seleccionada.

### 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>Opciones a elegir</title>
</head>
<body>
<html:form action="/administrarOpcion" method="POST">
<h1>Pantalla de opciones</h1>
<h2>Elige la operación que deseas</h2>

<html:select property="opcionElegida">
<html:option value="1">Consultar tus datos</html:option>
<html:option value="2">Modificar tus datos</html:option>
<html:option value="3">Consultar tus artículos</html:option>
</html:select>
<br><br>
<html:submit>Aceptar</html:submit>
</html:form>
</body>
</html>
```

El JavaBean que recibirá la opción elegida por el usuario es la clase **SeleccionarOpcionForm** que se encuentra en el paquete **ts.struts.javabeans**, el nombre que se indico en **property** del

tag <html:select> es “opcionElegida” por lo que el dato miembro que contenga el valor leído debe tener el mismo nombre

#### SeleccionarOpcionForm.java

```
package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class SeleccionarOpcionForm extends ActionForm{

    private int opcionElegida;

    public int getOpcionElegida() {
        return opcionElegida;
    }

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

}
```

Al ejecutarse la acción dentro del JSP, se llama al Action servlet referenciado por /administrarOpcion, es necesario desarrollar esa clase la cuál será llamada AdministrarOpcionAction y estará dentro del paquete ts.struts.servlets

#### AdministrarOpcionAction.java

```
package ts.struts.servlets;

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

import org.apache.struts.action.*;
import ts.struts.javabeans.*;

public class AdministrarOpcionAction extends Action{

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

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

        if(of.getOpcionElegida()==1)
            return mapping.findForward("consultarDatos");
        else if(of.getOpcionElegida()==2)
            return mapping.findForward("modificarDatos");
        else
            return mapping.findForward("consultarArticulos");
    }

}
```

Este llama a tres diferentes vistas (usando sus respectivas referencias) dependiendo el valor que se eligió en el menú de opciones, por lo que estos deben registrarse dentro de las acciones que realiza el servlet en el archivo **struts-config.xml**, al igual que el nuevo JavaBean utilizado, el **SeleccionarOpcionForm**.

#### **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>
  </global-forwards>
  <action-mappings>
    <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="request" type="ts.struts.servlets.ValidarUsuarioAction">
      <forward name="opciones" path="/paginas/opciones.jsp"></forward>
    </action>
    <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction">
      <forward name="consultarDatos"
path="/paginas/consultaDatos.jsp"></forward>
      <forward name="modificarDatos"
path="/paginas/modificaDatos.jsp"></forward>
      <forward name="consultarArticulos"
path="/paginas/consultaArticulos.jsp"></forward>
    </action>
  </action-mappings>
  <controller bufferSize="4096" debug="0" />
  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

Antes de ejecutar la aplicación, se deben desarrollar las tres vistas a la que se hace referencia en el archivo **struts-config.xml** y que serán llamadas dependiendo la acción elegida, por el momento estas no ofrecerán funcionalidad alguna, aunque se les agregarán dos ligas, una para salir del sistema (regresar a la pantalla de login) y otra para regresar al menú de opciones.

### consultaDatos.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 para consultar los datos</title>
</head>
<body>
<h1>Consulta de información</h1>
<h2>Tus datos son:</h2>
<center>
<html:link forward="opciones">Regresar al menú de
opciones</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>
</center>
</body>
</html>
```

### modificaDatos.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>Modificación de datos</title>
</head>
<body>
<h1>Modificación de datos</h1>
<center>
<html:link forward="opciones">Regresar al menú de
opciones</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>
</center>
</body>
</html>
```



## consultaArticulos.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>Consultar artículos</title>
</head>
<body>
<h1>Los artículos que tienes registrados son</h1>
<center>
<html:link forward="opciones"> Regresar al menú de
opciones</html:link><br><br>
<html:link forward="login"> Salir del sistema</html:link>
</center>
</body>
</html>
```

Las llamadas a las vistas referenciadas por “opciones” y “login” no forman parte de ningún ActionServlet, por lo que se deben declarar dentro de los campos de <global-forwards> en el archivo **struts-config.xml**.

Ya existe una referencia “opciones” dentro de las acciones del servlet **AdministrarOpcionAction**, al ser la misma que el global, esta se puede eliminar, de esta manera el archivo **struts-config.xml** queda de la siguiente manera

## 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="opciones" path="/paginas/opciones.jsp"></forward>
        <forward name="login" path="/paginas/login.jsp"></forward>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="request" type="ts.struts.servlets.ValidarUsuarioAction">
        </action>
        <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction">
        <forward name="consultarDatos"
```

```
path="/paginas/consultaDatos.jsp"></forward>
  <forward name="modificarDatos"
path="/paginas/modificaDatos.jsp"></forward>
  <forward name="consultarArticulos"
path="/paginas/consultaArticulos.jsp"></forward>
</action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

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

## JavaBeans accesibles durante la navegación

En la pantalla de validación de usuario, se introduce el login y el password, posteriormente se le pasa el control al ActionServlet ValidarUsuarioAction, el cual podría tener acceso a estos datos y pasárselos a la vista **opciones.jsp** que maneja el despliegue y lectura de la opción, sin embargo cuando ésta solicita a su ActionServlet (AdministrarOpcionAction), le pasa un JavaBean que consiste solamente en la opción elegida.

AdministrarOpcionAction debería conocer el login y el password para poder realizar una consulta en la base de datos y llenar la información del JSP a desplegar, ¿cómo hacer que la información del login y el password sean visibles para cualquier servlet?

Una manera es ir pasando esta información de JSP a un servlet y de este al siguiente JSP, sin embargo esto involucra crear variables en el código JSP, llenarlas y recuperarlas e incluso ocultar los campos en los JSP lo cuál puede llegar a resultar complicado y es fácil cometer errores.

Una forma más eficiente es indicar que un JavaBean estará disponible mientras una sesión esté activa. Para esto, en la declaración en el archivo **struts-config.xml** en donde se llena ese JavaBean, a la propiedad de **scope** se le debe asignar un valor de **session**. En este caso se desea que tanto el login como el password pertenecientes al bean **ValidarUsuarioForm** sean accesibles desde cualquier servlet o JSP.

```
<action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
</action>
```

### 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="opciones" path="/paginas/opciones.jsp"></forward>
    <forward name="login" path="/paginas/login.jsp"></forward>
  </global-forwards>
  <action-mappings>
    <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
    </action>
    <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction">
    <forward name="consultarDatos"
```

```

path="/paginas/consultaDatos.jsp"></forward>
<forward name="modificarDatos"
path="/paginas/modificaDatos.jsp"></forward>
<forward name="consultarArticulos"
path="/paginas/consultaArticulos.jsp"></forward>
</action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

De esta manera, en el ActionServlet **AdministrarOpcionAction** ya se puede tener acceso a ésta información a través de las siguientes líneas (es necesario importar la clase **javax.servlet.http.HttpSession**)

```

HttpSession sesion= request.getSession();
ValidarUsuarioForm vf =
(ValidarUsuarioForm) sesion.getAttribute("ValidarUsuarioForm");

```

Ahora el ActionServlet **AdministrarOpcionAction** puede conocer el valor de **login** y **password** para sus consultas y posterior llenado de las vistas correspondientes.

#### AdministrarOpcionAction.java

```

package ts.struts.servlets;

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

import org.apache.struts.action.*;
import ts.struts.javabeans.*;

public class AdministrarOpcionAction extends Action{

    public ActionForward execute(ActionMapping mapping,
                                ActionForm form,

    HttpServletRequest request,

    HttpServletResponse response) throws Exception{

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

        HttpSession sesion= request.getSession();

        ValidarUsuarioForm vf =
(ValidarUsuarioForm) sesion.getAttribute("ValidarUsuarioForm");

        System.out.println(vf.getLogin());
        System.out.println(vf.getPassword());
    }
}

```

```

        if(of.getOpcionElegida()==1)
            return mapping.findForward("consultarDatos");
        else if(of.getOpcionElegida()==2)
            return mapping.findForward("modificarDatos");
        else
            return mapping.findForward("consultarArticulos");
    }
}

```

Sin embargo, al finalizar la aplicación y volver a ejecutarla, los datos del usuario quedan almacenados, para solucionar esto, es necesario limpiar el objeto sesión en la vista de **login.jsp**, para que cada que se ingrese a esa pantalla, se tenga que introducir nuevamente un login y un password, esto se realiza con la instrucción

```
<%session.removeAttribute("nombre del JavaBean"); %>
```

### 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:html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Validación de usuario</title>
</head>
<body>
<%session.removeAttribute("ValidarUsuarioForm"); %>
<html:form action="/validarUsuario" method="POST">
<center>
<h1>Introduce tus datos</h1>
Usuario:
<html:text property="login" ></html:text>
<br><br>
Password:
<html:password property="password"></html:password>
<br><br>
<html:submit>Aceptar</html:submit>
<html:reset>Limpiar</html:reset>
</center>
</html:form>
</body>
</html:html>

```

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

## 2. Consulta de los datos del usuario

Lo primero es crear una clase que contenga los datos del usuario a manejar en el sistema.

Crear una clase llamada **DatosUsuarioForm** en el paquete **ts.struts.javabeans**, al ser un **JavaBean**, debe heredar de la clase **ActionForm** con los datos miembro para: nombre, apellido paterno, apellido materno, teléfono, e-mail, RFC, login y password.

### **DatosUsuarioForm.java**

```
package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class DatosUsuarioForm extends ActionForm{

    private String nombre;
    private String apellidoPaterno;
    private String apellidoMaterno;
    private String telefono;
    private String email;
    private String RFC;
    private String login;
    private String password;

    public String getNombre() {
        return nombre;
    }
    public String getApellidoPaterno() {
        return apellidoPaterno;
    }
    public String getApellidoMaterno() {
        return apellidoMaterno;
    }
    public String getTelefono() {
        return telefono;
    }
    public String getEmail() {
        return email;
    }
    public String getRFC() {
        return RFC;
    }
    public String getLogin() {
        return login;
    }
    public String getPassword() {
        return password;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setApellidoPaterno(String apellidoPaterno) {
        this.apellidoPaterno = apellidoPaterno;
    }
}
```

```

    public void setApellidoMaterno(String apellidoMaterno) {
        this.apellidoMaterno = apellidoMaterno;
    }
    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void setRFC(String rfc) {
        RFC = rfc;
    }
    public void setLogin(String login) {
        this.login = login;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}

```

Al ser un JavaBean, debe registrarse en el archivo **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="DatosUsuarioForm"
type="ts.struts.javabeans.DatosUsuarioForm"></form-bean>
    </form-beans>
    <global-exceptions />
    <global-forwards>
        <forward name="opciones" path="/paginas/opciones.jsp"></forward>
        <forward name="login" path="/paginas/login.jsp"></forward>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
        </action>
        <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction">
        <forward name="consultarDatos"
path="/paginas/consultaDatos.jsp"></forward>
        <forward name="modificarDatos"
path="/paginas/modificaDatos.jsp"></forward>
        <forward name="consultarArticulos"
path="/paginas/consultaArticulos.jsp"></forward>
        </action>
    </action-mappings>
    <controller bufferSize="4096" debug="0" />
    <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

Lo siguiente es desarrollar las clases que obtengan la información de una base de datos (por el momento simulada), esta clase se encontrará en el paquete **ts.struts.modelo**. El nombre de la clase es **ObtenerInformacionUsuario**

#### ObtenerInformacionUsuario.java

```
package ts.struts.modelo;

import ts.struts.javabeans.*;

public class ObtenerInformacionUsuario {

    public DatosUsuarioForm obtenerDatosUsuario(String login){

        DatosUsuarioForm uf = new DatosUsuarioForm();

        if(login.compareTo("josue")==0){

            uf.setNombre("Josué");
            uf.setApellidoPaterno("Figueroa");
            uf.setApellidoMaterno("González");
            uf.setEmail("josue.figueroa@gmail.com");
            uf.setTelefono("57899810");
            uf.setRFC("FIGJ780718");
            uf.setLogin(login);
            uf.setPassword("password");

        }

        else {

            uf.setNombre("Blanca Estela");
            uf.setApellidoPaterno("Sánchez");
            uf.setApellidoMaterno("Valencia");
            uf.setEmail("besv@hotmail.com");
            uf.setTelefono("1234567");
            uf.setRFC("SAVB770602");
            uf.setLogin(login);
            uf.setPassword("password");

        }

        return uf;

    }

}
```

Modificar la clase **AdministrarOpcionAction** para que realice la consulta y despliegue los datos, para realizar el despliegue es necesario pasar el JavaBean a una variable que se encuentre en el JSP

```
request.setAttribute("datos", uf);
```

Así se indica que la variable **uf** (que contiene un objeto DatosUsuarioForm) se le pasa a la variable llamado **datos** en el JSP



## AdministrarOpcionAction.java

```
package ts.struts.servlets;

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

import org.apache.struts.action.*;
import ts.struts.javabeans.*;
import ts.struts.modelo.*;

public class AdministrarOpcionAction extends Action{

    public ActionForward execute(ActionMapping mapping,
                                ActionForm form,

                                HttpServletRequest request,

                                HttpServletResponse response) throws Exception{

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

        HttpSession sesion= request.getSession();

        ValidarUsuarioForm vf =
        (ValidarUsuarioForm) sesion.getAttribute("ValidarUsuarioForm");

        System.out.println(vf.getLogin());
        System.out.println(vf.getPassword());

        if(of.getOpcionElegida()==1){

            DatosUsuarioForm uf = new DatosUsuarioForm();
            ObtenerInformacionUsuario obtener = new
            ObtenerInformacionUsuario();
            uf=obtener.obtenerDatosUsuario(vf.getLogin());
            request.setAttribute("datos",uf);
            return mapping.findForward("consultarDatos");
        }

        else if(of.getOpcionElegida()==2)
            return mapping.findForward("modificarDatos");
        else
            return mapping.findForward("consultarArticulos");
    }
}
```

Como el atributo `datos` usado en la función `request.setAttribute("datos",uf);` es un `JavaBean` registrado en el archivo `struts-config.xml`, puede imprimirse haciendo uso de los tags de manejo de beans, para tener acceso a estos tags es necesario incluir el siguiente código en la vista `consultaDatos.jsp`

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

De esta manera, la variable **"datos"** tiene todos los datos miembro del JavaBean **DatosUsuarioForm** por lo que la impresión de alguno de sus datos sería:

```
<bean:write name="datos" property="nombre del atributo"/><br><br>
```

### consultaDatos.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>Página para consultar los datos</title>
</head>
<body>
<h1>Consulta de información</h1>
<h2>Tus datos son:</h2>
Nombre: <bean:write name="datos" property="nombre"/><br><br>
Apellido paterno: <bean:write name="datos"
property="apellidoPaterno"/><br><br>
Apellido materno: <bean:write name="datos"
property="apellidoMaterno"/><br><br>
Teléfono: <bean:write name="datos" property="telefono"/><br><br>
RFC: <bean:write name="datos" property="RFC"/><br><br>
<br>
Login: <bean:write name="datos" property="login"/><br><br>
Password: <bean:write name="datos" property="password"/><br><br>
<center>
<html:link forward="opciones">Regresar al menú de
opciones</html:link><br><br>
<html:link forward="login">Salir del sistema</html:link>
</center>
</body>
</html>
```

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

## Varios métodos de ejecución

Hasta el momento, se ha implementado de forma sencilla la funcionalidad de desplegar los datos de un usuario, esto se realizó en el método `execute()` de la clase **AdministrarOpcionAction**.

Faltando dos funcionalidades por implementar se puede observar que el código de éste método puede crecer en tamaño y complejidad en caso de que se manejen más opciones.

Por esta razón antes de implementar el resto de las funcionalidades, se estudiará una forma de que haya un método para cada una de las opciones que se puedan elegir.

Esto es posible con la clase `DispatchAction()`

Esta clase permite definir en la misma clase diferentes métodos para tratar un grupo de peticiones similares. Su uso en el servlet es el siguiente:

```
public class Clase extends DispatchAction {

    public ActionForward nombre_1 (ActionMapping mapping, ActionForm form

                                HttpServletRequest request,

                                HttpServletResponse response)

                                Throws Exception{

                                //IMPLEMENTACIÓN

                                }

    public ActionForward nombre_2 (ActionMapping mapping, ActionForm form

                                HttpServletRequest request,

                                HttpServletResponse response)

                                Throws Exception{

                                //IMPLEMENTACIÓN

                                }

}
```

Su definición dentro del archivo struts-config.xml es agregando el campo **parameter** a la definición de la acción del ActionServlet

```
<action path="/alias" type="clase" parameter="atributo">
```

En donde atributo es el nombre del atributo que recibe la opción elegida

En el JSP, suponiendo que se tiene una lista de selección, se tendría lo siguiente:

```
<html: select property="atributo">
```

```
<html:option value="valor">Texto</html:option>
```

Aquí es importante que el texto colocado en "valor" sea igual al nombre del método al que se llamará.

## Implementación.

Se comenzará cambiando el código de la clase **AdministrarOpcionAction** para que ésta herede de **DispatchAction** en lugar de **Action** siendo necesario realizar la siguiente importación

```
import org.apache.struts.actions.DispatchAction;
```

Ahora se crearán tres métodos (ya que aquí se manejan tres opciones):

- **consultarDatosUsuario**
- **modificarDatosUsuario**
- **consultarArticulos**

Con esto es posible eliminar el método **execute()**

### AdministrarOpcionAction.java

```
package ts.struts.servlets;

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

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

import ts.struts.javabeans.*;
import ts.struts.modelo.*;

public class AdministrarOpcionAction extends DispatchAction{
```

```

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

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

    HttpSession sesion= request.getSession();
    ValidarUsuarioForm vf =
    (ValidarUsuarioForm) sesion.getAttribute("ValidarUsuarioForm");
    DatosUsuarioForm uf = new DatosUsuarioForm();

    ObtenerInformacionUsuario obtener = new
    ObtenerInformacionUsuario();

    uf=obtener.obtenerDatosUsuario(vf.getLogin());
    request.setAttribute("datos",uf);

    return mapping.findForward("consultarDatos");
}

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

    return mapping.findForward("modificarDatos");
}

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

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

```

El siguiente paso es modificar la vista en donde se lee la opción elegida, **opciones.jsp**, en este caso, el nombre del atributo (campo **property**) que recibe la opción es **"opcionElegida"**, y los posibles valores que puede tomar son "1", "2" o "3" estos valores deben coincidir con los nombres de los métodos por lo que es necesario modificarlos.

#### 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>Opciones a elegir</title>
</head>

```

```

<body>
<html:form action="/administrarOpcion" method="POST">
<h1>Pantalla de opciones</h1>
<h2>Elige la operación que deseas</h2>

<html:select property="opcionElegida">
<html:option value="mostrarDatosUsuario">Consultar tus
datos</html:option>
<html:option value="modificarDatosUsuario">Modificar tus
datos</html:option>
<html:option value="consultarArticulos">Consultar tus
artículos</html:option>
</html:select>
<br><br>
<html:submit>Aceptar</html:submit>
</html:form>
</body>
</html>

```

Sin embargo, en el JavaBean leído (**SeleccionarOpcionForm**), el atributo **opcionElegida** es un entero (**int**), esto ya no coincide con el valor leído actualmente, será necesario modificarlo a que sea una cadena (**String**)

#### SeleccionarOpcionForm.java

```

package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class SeleccionarOpcionForm extends ActionForm{

    private String opcionElegida;

    public String getOpcionElegida() {
        return opcionElegida;
    }

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

}

```

Solo falta registrar estas modificaciones en el archivo **struts-config.xml**, esto es en las acciones realizadas por **AdministrarOpcionAction**, agregando el campo **parameter="nombre"**, donde **nombre** se refiere al nombre del atributo que recibe la opción, en este caso **"opcionElegida"**

#### 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="DatosUsuarioForm"
type="ts.struts.javabeans.DatosUsuarioForm"></form-bean>
</form-beans>
<global-exceptions />
<global-forwards>
<forward name="opciones" path="/paginas/opciones.jsp"></forward>
<forward name="login" path="/paginas/login.jsp"></forward>
</global-forwards>
<action-mappings>
<action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
</action>
<action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction"
parameter="opcionElegida">
<forward name="consultarDatos"
path="/paginas/consultaDatos.jsp"></forward>
<forward name="modificarDatos"
path="/paginas/modificaDatos.jsp"></forward>
<forward name="consultarArticulos"
path="/paginas/consultaArticulos.jsp"></forward>
</action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

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

### 3. Consultas de artículos pertenecientes a un usuario

Se realizará la consulta de todos los artículos que pertenecen a un determinado usuario, consultando todos sus artículos y posteriormente utilizando uno o más criterios de búsqueda.

Consulta de todos los artículos.

- Crear una clase llamada **InformacionArticuloForm** en el paquete **ts.struts.javabeans**
- No es necesario registrarla en el archivo **struts-config.xml**

**InformacionArticuloForm.java**

```
package ts.struts.javabeans;

import org.apache.struts.action.ActionForm;

public class InformacionArticuloForm extends ActionForm{

    private String tipo;
    private String nombre;
    private String genero;
    private String annio;

    public String getTipo() {
        return tipo;
    }
    public String getNombre() {
        return nombre;
    }
    public String getGenero() {
        return genero;
    }
    public String getAnnio() {
        return annio;
    }
    public void setTipo(String tipo) {
        this.tipo = tipo;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setGenero(String genero) {
        this.genero = genero;
    }
    public void setAnnio(String annio) {
        this.annio = annio;
    }
}
```

Crear una clase que contenga un tipo colección que almacene la lista de los elementos, en este caso la colección será de tipo LinkedList (Lista Ligada).

- Crear una clase llamada **ListaArticulosForm** en el paquete **ts.struts.javabeans**
- Registrarla en el archivo **struts-config.xml**



## ListaArticulosForm.java

```
package ts.struts.javabeans;

import java.util.LinkedList;

import org.apache.struts.action.ActionForm;

public class ListaArticulosForm extends ActionForm{

    private LinkedList lista;

    public LinkedList getLista() {
        return lista;
    }

    public void setLista(LinkedList lista) {
        this.lista = lista;
    }
}
```

## 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="DatosUsuarioForm"
type="ts.struts.javabeans.DatosUsuarioForm"></form-bean>
        <form-bean name="ListaArticulosForm"
type="ts.struts.javabeans.ListaArticulosForm"></form-bean>
    </form-beans>
    <global-exceptions />
    <global-forwards>
        <forward name="opciones" path="/paginas/opciones.jsp"></forward>
        <forward name="login" path="/paginas/login.jsp"></forward>
    </global-forwards>
    <action-mappings>
        <action path="/validarUsuario" name="ValidarUsuarioForm"
scope="session" type="ts.struts.servlets.ValidarUsuarioAction">
        </action>
        <action path="/administrarOpcion" name="SeleccionarOpcionForm"
scope="request" type="ts.struts.servlets.AdministrarOpcionAction"
parameter="opcionElegida">
        <forward name="consultarDatos"
path="/paginas/consultaDatos.jsp"></forward>
        <forward name="modificarDatos"
path="/paginas/modificaDatos.jsp"></forward>
        <forward name="consultarArticulos"
path="/paginas/consultaArticulos.jsp"></forward>
        <forward name="consultaAvanzada"
path="/paginas/consultaArticulosAvanzada.jsp"></forward>
```

```

</action>
</action-mappings>
<controller bufferSize="4096" debug="0" />
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

En el paquete **ts.struts.modelo** se creará una clase llamada **ObtenerArticulosUsuario** encargada de simular la recuperación de los artículos y la creación de un objeto tipo **ListaArticulosForm**

#### ObtenerArticulosUsuario.java

```

package ts.struts.modelo;

import java.util.LinkedList;

import ts.struts.javabeans.InformacionArticuloForm;
import ts.struts.javabeans.ListaArticulosForm;

public class ObtenerArticulosUsuario {

    public ListaArticulosForm llenarListaDeArticulos(String login){

        ListaArticulosForm listaArticulos = new
        ListaArticulosForm();

        LinkedList aux = new LinkedList();

        InformacionArticuloForm articulo_1 = new
        InformacionArticuloForm();
        InformacionArticuloForm articulo_2 = new
        InformacionArticuloForm();
        InformacionArticuloForm articulo_3 = new
        InformacionArticuloForm();

        if(login.compareTo("josue")==0){
            articulo_1.setTipo("Audio");
            articulo_1.setGenero("Clasica");
            articulo_1.setNombre("La 9na sinfonia");
            articulo_1.setAnio("2008");

            articulo_2.setTipo("Video");
            articulo_2.setGenero("Terror");
            articulo_2.setNombre("Viernes 13");
            articulo_2.setAnio("2009");

            articulo_3.setTipo("Video juego");
            articulo_3.setGenero("Accion");
            articulo_3.setNombre("God of War 2");
            articulo_3.setAnio("2007");

            aux.add(articulo_1);
            aux.add(articulo_2);
            aux.add(articulo_3);
        }
    }
}

```

```

        else {
            articulo_1.setTipo("Audio");
            articulo_1.setGenero("Rock");
            articulo_1.setNombre("Bon Jovi");
            articulo_1.setAnio("2006");

            articulo_2.setTipo("Video");
            articulo_2.setGenero("Accion");
            articulo_2.setNombre("Spider-Man 2");
            articulo_2.setAnio("2007");

            articulo_3.setTipo("Video juego");
            articulo_3.setGenero("Musica");
            articulo_3.setNombre("Rock Ballads");
            articulo_3.setAnio("2008");

            aux.add(articulo_1);
            aux.add(articulo_2);
            aux.add(articulo_3);
        }
        listaArticulos.setLista(aux);
        return listaArticulos;
    }
}

```

Modificar la clase **AdministrarOpcionAction** para que obtenga la lista de artículos y los envíe a la vista correspondiente. El método que mandaba a desplegar la lista de artículos es **consultarArticulos()** por lo que será el único que se modifique. Como se enviará un JavaBean a una variable en un JSP es necesario utilizar la instrucción

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

En donde lista es un objeto tipo **ListaArticulosForm** que se le pasará a una variable llamada **listaArticulos** en el JSP. En el método también se incluye código para recuperar el login y el password de la sesión (igual que en la consulta de datos) para solicitar los artículos asociados a un login. El código del método **listaArticulos()** dentro de la clase **AdministrarOpcionAction** es el siguiente:

```

public ActionForward consultarArticulos(ActionMapping mapping,
                                       ActionForm form,
                                       HttpServletRequest request,
                                       HttpServletResponse response) throws Exception{
    HttpSession sesion= request.getSession();
    ValidarUsuarioForm vf =
    (ValidarUsuarioForm) sesion.getAttribute("ValidarUsuarioForm");

    ListaArticulosForm lista = new ListaArticulosForm();
    ObtenerArticulosUsuario obtener = new
    ObtenerArticulosUsuario();
    lista = obtener.llenarListaDeArticulos(vf.getLogin());
    request.setAttribute("listaArticulos", lista);
    return mapping.findForward("consultarArticulos");
}

```

## Desplegando la lista en un JSP

Como del ActionServlet **AdministrarOpcionAction** se recibe un JavaBean, para manejarlo en el JSP es necesario incluir el código:

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

Se realizará una operación lógica de iteración para recorrer la lista, para tener acceso a esta operación, se debe incluir la taglib necesaria a través de la instrucción:

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

Esta taglib permite el uso de <logic:iterate>, su funcionamiento es el siguiente:

```
<logic iterate name="nombre" porperty="dato miembro" id="identificador">
```

//CÓDIGO

```
</logic:iterate>
```

En donde:

nombre representa el nombre de la variable tipo JavaBean que se le envió a través del ActionServlet

dato miembro representa el dato miembro (de tipo colección) que se encuentra en la variable nombre

identificador representa un identificador que se le dará a los elementos que se encuentren dentro de la colección identificada por **dato miembro**

De esta manera, en el ejemplo se tiene que el nombre del JavaBean que contiene la lista y que se le pasó al JSP a través del ActionServlet es **listaArticulos**, dentro de ese JavaBean se encuentra un atributo tipo LinkedList llamado **lista** y se identificará al artículo con el nombre **articulo**, por lo que el código quedaría de la siguiente manera:

```
<logic:iterate name="listaArticulos" property="lista" id="articulo">
```

Para imprimir los datos del artículo, se utiliza la instrucción <bean:write> de manera similar como se utilizó para imprimir los datos del usuario, pero aquí el identificador que tiene la información del artículo (tipo InformacionArticuloForm) de acuerdo a la declaración en logic:iterate es **"articulo"** por lo que la impresión de uno de sus datos miembro sería

```
<bean:write name="articulo" property="tipo"/>
```

consultarArticulo.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>Consultar artículos</title>
</head>
<body>
<h1>Los artículos que tienes registrados 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="lista" id="articulo">
<tr>
<td><bean:write name="articulo" property="tipo"/></td>
<td><bean:write name="articulo" property="nombre"/></td>
<td><bean:write name="articulo" property="genero"/></td>
<td><bean:write name="articulo" property="annio"/></td>
</tr>
</logic:iterate>
</table>
</center>
<br><br>
<center>
<html:link forward="opciones"> Regresar al menú de
opciones</html:link><br><br>
<html:link forward="login"> Salir del sistema</html:link>
</center>
</body>
</html>
```

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