Laboratorio. Desplegando listas.

Objetivo.
Realizar una aplicación que permita desplegar una lista de elementos, en donde esos elementos pueden llegar a ser listas de elementos.

Actividades a realizar.

1. Crear un nuevo proyecto.
2. Realizar una vista que permita seleccionar entre desplegar una lista o una lista de listas.
3. Realizar la vista y las clases necesarias para desplegar una lista.
4. Realizar la vista y las clases necesarias para desplegar una lista de listas
5. Desplegar una lista utilizando DisplayTag

1. Crear un nuevo proyecto.

- Crear y configurar un nuevo entorno de trabajo.
- Crear los siguientes paquetes.
    - **ts.struts.servlets**
    - **ts.struts.modelo**
    - **ts.struts.beans**

2. Vista para la elección del tipo de lista a desplegar.

Crear y registrar en el archivo **struts-config.xml**, el bean que reciba la opción seleccionada en el paquete **ts.struts.beans**, en este caso solo es un bean con un atributo String que reciba la opción elegida.

ManejoOpcionForm.java

```
package ts.struts.beans;
import org.apache.struts.action.ActionForm;

public class ManejoOpcionForm extends ActionForm{

	private String opcionElegida;

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

La vista para el menú es:

menu.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>Selecciona la opciÃ³n</title>
</head>
<body>
<html:form action="/administrarMenu" method="POST">
<html:select property="opcionElegida">
<html:option value="unaLista">Desplegar una sola lista</html:option>
<html:option value="listaDeListas">Desplegar una lista de listas</html:option>
</html:select>
<br><br>
<html:submit>Aceptar</html:submit>
</html:form>
</body>
</html>
```

Esta vista hace una llamada al ActionServlet referenciado por "/administrarMenu", la clase que le corresponde se crea en el paquete **ts.struts.servlets**. Se desea que se tenga un método por cada una de las opciones disponibles. Este ActionServlet debe ser registrado en el archivo **struts-config.xml**.

AdministrarMenuAction.java

```
package ts.struts.servlets;

import java.util.LinkedList;

import javax.servlet.http.*;

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

import ts.struts.beans.*;
import ts.struts.modelo.*;

public class AdministrarMenuAction extends DispatchAction{

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

            EmpleadosDepartamentoForm df = new EmpleadosDepartamentoForm();
            DatosDepartamento manejoDpto = new DatosDepartamento();
```

```
            LlenarDepartamento llenar = new LlenarDepartamento();
            manejoDpto = llenar.llenarLista();
            df.setDepartamento(manejoDpto);

            request.setAttribute("depto",df);

            return mapping.findForward("listaSencilla");
      }

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

            EmpleadosDivisionForm df = new EmpleadosDivisionForm();
            DatosDivision manejoDiv = new DatosDivision();
            LlenarDivision llenar = new LlenarDivision();

            manejoDiv = llenar.llenar();
            df.setDivision(manejoDiv);

            request.setAttribute("div",df);

            return mapping.findForward("listaDeListas");
      }

}
```

Registro en **struts-config.xml**

```
<action     path="/administrarMenu"     name="ManejoOpcionForm"     scope="request"
type="ts.struts.servlets.AdministrarMenuAction" parameter="opcionElegida">
<forward name="listaDeListas" path="/paginas/listaDeListas.jsp"></forward>
<forward name="listaSencilla" path="/paginas/listaSencilla.jsp"></forward>
</action>
```

Cada método llama a una vista diferente dependiendo lo que se desea desplegar.

3. Desplegar una lista sencilla

Lo primero es crear una clase que llene la lista, en este caso se desea desplegar una lista que contenga los datos de los profesores que laboran en un cierto departamento, cada uno de esos profesores tiene determinados atributos-

En el paquete **ts.struts.modelo** se creará la clase que contenga los datos de los profesores.

DatosEmpleado.java

```
package ts.struts.modelo;

public class DatosEmpleado {

	private String nombre;
	private String apellidoPaterno;
	private String apellidoMaterno;
	private String id;

	public DatosEmpleado(String n, String ap, String am, String i){
		this.nombre=n;
		this.apellidoPaterno=ap;
		this.apellidoMaterno=am;
		this.id=i;
	}

	public String getNombre() {
		return nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public String getApellidoPaterno() {
		return apellidoPaterno;
	}
	public void setApellidoPaterno(String apellidoPaterno) {
		this.apellidoPaterno = apellidoPaterno;
	}
	public String getApellidoMaterno() {
		return apellidoMaterno;
	}
	public void setApellidoMaterno(String apellidoMaterno) {
		this.apellidoMaterno = apellidoMaterno;
	}
	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}

}
```

Es necesaria una clase que contenga los datos del departamento, por el momento solo se considera el nombre y una lista de los empleados, esta clase se creará en el paquete **ts.struts.modelo**.

DatosDepartamento.java

```
package ts.struts.modelo;

import java.util.LinkedList;

public class DatosDepartamento {

	private String nombre;
	private LinkedList listaEmpleados;

	public String getNombre() {
		return nombre;
	}
	public LinkedList getListaEmpleados() {
		return listaEmpleados;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public void setListaEmpleados(LinkedList listaEmpleados) {
		this.listaEmpleados = listaEmpleados;
	}
}
```

Por el momento solo se simulará que se obtienen los datos de los empleados de una base de datos, por lo que es necesario crear una clase que haga el llenado de la lista de empleados, esta clase se encuentra en el paquete **ts.struts.modelo.**

LlenarDepartamento.java

```
package ts.struts.modelo;

import java.util.LinkedList;
import ts.struts.beans.EmpleadosDepartamentoForm;

public class LlenarDepartamento {


	public DatosDepartamento llenarLista(){

		DatosDepartamento departamento=new DatosDepartamento();
		LinkedList empleados = new LinkedList();
		empleados.add(new DatosEmpleado("Josue","Figueroa","Gonzalez","123"));
		empleados.add(new
DatosEmpleado("Ivonne","Figueroa","Gonzalez","456"));
		empleados.add(new DatosEmpleado("Blanca","Sanchez","Valencia","789"));
		departamento.setListaEmpleados(empleados);
		departamento.setNombre("Sistemas");

		return departamento;
	}
}
```

Crear el ActionForm que contenga los datos a desplegar en pantalla, para simplificar su contenido se puede indicar que el único dato miembro es un objeto de otra clase que contenga la información, en este caso, el contenido de un departamento es de tipo DatosDepartamento, de esta manera en el paquete ts.struts.beans se crea la clase EmpleadosDepartamentoForm.

EmpleadosDepartamentoForm.java

```
package ts.struts.beans;

import java.util.LinkedList;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

import ts.struts.modelo.DatosDepartamento;
import ts.struts.modelo.DatosEmpleado;

public class EmpleadosDepartamentoForm extends ActionForm{

	DatosDepartamento departamento;

	public DatosDepartamento getDepartamento() {
		return departamento;
	}

	public void setDepartamento(DatosDepartamento departamento) {
		this.departamento = departamento;
	}

}
```

Este bean debe ser registrado en el archivo **struts-config.xml**

Este bean se le pasa como atributo a la vista en el ActionServlet a través de la instrucción

```
request.setAttribute("depto",df);
```

Para el despliegue de la lista se utilizará el taglib logic, en particular logic:iterate

listaSencilla.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 elementos</title>
</head>
<body>
<h1>Los empleados del departamento de
<bean:write name="depto"
property="departamento.nombre"></bean:write></h1><br><br>
<logic:iterate name="depto"property="departamento.listaEmpleados" id="empleado">
<bean:write name="empleado" property="id"></bean:write>
<bean:write name="empleado" property="nombre"></bean:write>
<bean:write name="empleado" property="apellidoPaterno"></bean:write>
<bean:write name="empleado" property="apellidoMaterno"></bean:write><br>
</logic:iterate>
<br><br>
</body>
<html:link forward="menu">Presiona para regresar al menu</html:link>
</html>
```

4. Desplegar una lista de listas.

Se desea una vista que despliegue una lista de listas, en este caso los datos de una división, que contiene un nombre y una lista de los departamentos que se encuentran en ella, cada uno de esos departamentos contiene una lista de los profesores que laboran en ellos.

Crear una clase en el paquete **ts.struts.modelo** que represente la información que se tiene en una división.

DatosDivision.java

```
package ts.struts.modelo;

import java.util.LinkedList;

public class DatosDivision {

	private String nombre;
	private LinkedList departamentos;

	public String getNombre() {
		return nombre;
	}
	public LinkedList getDepartamentos() {
		return departamentos;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public void setDepartamentos(LinkedList departamentos) {
		this.departamentos = departamentos;
	}
}
```

Crear una clase en el paquete **ts.struts.modelo** que permita llenar la información de una División

LlenarDivision.java

```
package ts.struts.modelo;

import java.util.LinkedList;

public class LlenarDivision {

	public DatosDivision llenar(){

		LinkedList listaDepartamentos = new LinkedList();
		LinkedList empleados;
		DatosDepartamento departamento;
		DatosDivision division=new DatosDivision();

		empleados=new LinkedList();
		departamento = new DatosDepartamento();

		empleados.add(new DatosEmpleado("Josue","Figueroa","Gonzalez","123"));
		empleados.add(new
DatosEmpleado("Ivonne","Figueroa","Gonzalez","456"));
		empleados.add(new DatosEmpleado("Blanca
Estela","Sanchez","Valencia","789"));


		departamento.setListaEmpleados(empleados);
		departamento.setNombre("Sistemas");
		listaDepartamentos.add(departamento);

		empleados=new LinkedList();
		departamento = new DatosDepartamento();

		empleados.add(new
DatosEmpleado("Nombre1","Paterno1","Materno1","321"));
		empleados.add(new
DatosEmpleado("Nombre2","Paterno2","Materno2","654"));
		empleados.add(new
DatosEmpleado("Nombre3","Paterno3","Materno3","987"));

		departamento.setListaEmpleados(empleados);
		departamento.setNombre("Electronica");
		listaDepartamentos.add(departamento);

		division.setDepartamentos(listaDepartamentos);
		division.setNombre("CBI");

		return division;

	}

}
```

Es necesario crear un bean que contenga la información que se desplegará en pantalla, en este caso esa información se encuentra en la clase DatosDivision, por lo que el único dato miembro necesario es uno de ese tipo, éste bean se crea en el paquete **ts.struts.beans**.

EmpleadosDivisionForm.java

```
package ts.struts.beans;

import java.util.LinkedList;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import ts.struts.modelo.*;

public class EmpleadosDivisionForm extends ActionForm{

	DatosDivision division;

	public DatosDivision getDivision() {
		return division;
	}

	public void setDivision(DatosDivision division) {
		this.division = division;
	}
}
```

Desarrollo de la vista.

En el ActionServlet se envía la información a un atributo llamado “div” a través de la instrucción:

```
request.setAttribute("div",df);
```

Para el despliegue de una lista de listas, se deben utilizar dos iteraciones anidadas, una para recorrer los departamentos y otra para recorrer los empleados del departamento.

Lo primero es escribir el nombre de la división que se encuentra en el atributo “div”

```
<h1>Los empleados del departamento de
<bean:write name="div" property="division.nombre"></bean:write></h1>
```

Lo siguiente es iterar dentro de los departamentos que conforman la división

```
<logic:iterate name="div" property="division.departamentos" id="depto">
```

Esto indica que dentro del atributo “div” se encuentra un atributo llamado division y que dentro de esa división se encuentra un atributo tipo colección (LinkedList) llamado departamentos, cada uno de los cuáles se identificará a través del nombre “depto”.

Con las siguientes líneas se escribe el nombre del departamento:

```
<h2>Departamento de :
<bean:write name="depto" property="nombre"></bean:write></h2>
```

Posteriormente se debe iterar dentro del objeto “depto”, el cuál es de tipo DatosDepartamento y contiene un atributo tipo colección (LinkedList) que tiene dentro objetos de tipo DatosEmpleado, cada uno de los cuáles se identificará con el nombre “empleado”

```
<logic:iterate name="depto"property="listaEmpleados" id="empleado">
```

Finalmente solo hay que desplegar los datos de los empleados, por ejemplo:

```
<bean:write name="empleado" property="nombre"></bean:write>
```

De esta manera el código de la vista queda de la siguiente manera:

listaDeListas.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://displaytag.sf.net" prefix="display" %>
<%@ 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 listas</title>
</head>
<body>
<h1>Los empleados del departamento de
<bean:write name="div" property="division.nombre"></bean:write></h1>
<logic:iterate name="div" property="division.departamentos" id="depto">
<h2>Departamento de :
<bean:write name="depto" property="nombre"></bean:write></h2>
<logic:iterate name="depto"property="listaEmpleados" id="empleado">
<bean:write name="empleado" property="id"></bean:write>
<bean:write name="empleado" property="nombre"></bean:write>
<bean:write name="empleado" property="apellidoPaterno"></bean:write>
<bean:write name="empleado" property="apellidoMaterno"></bean:write><br>
</logic:iterate>
</logic:iterate>
</body>
```

```
<br><br>
<html:link forward="menu">Presiona para regresar al menu</html:link>
</html>
```

El contenido del archivo struts-config.xml es el siguiente:

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="EmpleadosDepartamentoForm"
type="ts.struts.beans.EmpleadosDepartamentoForm"></form-bean>
   <form-bean name="EmpleadosDivisionForm"
type="ts.struts.beans.EmpleadosDivisionForm"></form-bean>
   <form-bean name="ManejoOpcionForm"
type="ts.struts.beans.ManejoOpcionForm"></form-bean>
   </form-beans>
   <global-exceptions />
   <global-forwards>
   <forward name="menu" path="/paginas/menu.jsp"></forward>
   </global-forwards>
   <action-mappings>
   <action path="/administrarMenu" name="ManejoOpcionForm" scope="request"
type="ts.struts.servlets.AdministrarMenuAction" parameter="opcionElegida">
   <forward name="listaDeListas" path="/paginas/listaDeListas.jsp"></forward>
   <forward name="listaSencilla" path="/paginas/listaSencilla.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 menu.jsp en el servidor

5. Desplegar una lista usando la biblioteca Display-tag

La librería displaytag proporciona un conjunto de operaciones que facilitan el despliegue de información entre otras características.

Para utilizarla es necesario descargar e instalar las bibliotecas necesarias
.
Para su instalación, es necesario:

- Descargar la biblioteca display de http://www.display.org
- Descargar una versión 2.0 o superior de commons-lang http://jakarta.apache.org/commons/lang
- Extraer los archivos
- Copiar el archivo displaytag-1.0.jar del archivo al directorio /WEB-INF/lib/ del proyecto
- Copair el archivo displaytag-11.tld al directorio WEB-INF del proyecto
- Copiar el archivo commons-lang-2.X.jar al directorio /WEB-INF/lib/
- Copiar el archivo commons-collections-2.1.1.jar al directorio /WEB-INF/lib del proyecto

Creación de la vista

Solo es necesario modificar la vista, los ActionServlets, ActionForms y clases del modelo siguen funcionando ya que al final se le pasa un bean que contiene entre sus elementos una lista. Se creará la vista para desplegar SOLAMENTE UNA lista.

Para poder utilizar las operaciones de la tag display, es necesario incluir la siguiente instrucción en el código de la página JSP.

```
<%@ taglib uri="http://displaytag.sf.net" prefix="display" %>
```

Considerar que el ActionServlet que solicita a ala vista le envió la información del bean (de tipo EmpleadosDepartamentoForm) a la vista a través del atributo “depto”.

Para desplegar la tabla se tiene que indicar cuál de los datos miembro contiene la tabla a utilizar y posteriormente desplegarla.

```
<display:table name="depto.departamento.listaEmpleados">
<display:column property="id" title="ID"></display:column>
<display:column property="nombre" title="Nombre"></display:column>
<display:column property="apellidoPaterno" title="Apellido
paterno"></display:column>
<display:column property="apellidoMaterno" title="Apellido
materno"></display:column>
</display:table><br>
```

Estas instrucciones indican que se trabajará con el dato depto.departamento.listaEmpleados que al final es la que contiene la lista de los empleados que pertenecen a un departamento. Posteriormente se puede personalizar el encabezado de una columna a través de la instrucción:

```
<display:column property="nombre" title="Nombre"></display:column>
```

En donde “nombre” corresponde a un atributo de un empleado, que son los datos que se encuentran en la lista “listaEmpleados”.

El código completo de la vista es:

listaConTags.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://displaytag.sf.net" prefix="display" %>
<!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 elementos</title>
</head>
<body>
<h1>Los empleados del departamento de
<bean:write name="depto"
property="departamento.nombre"></bean:write></h1><br><br>
<display:table name="depto.departamento.listaEmpleados">
<display:column property="id" title="ID"></display:column>
<display:column property="nombre" title="Nombre"></display:column>
<display:column property="apellidoPaterno" title="Apellido
paterno"></display:column>
<display:column property="apellidoMaterno" title="Apellido
materno"></display:column>
</display:table><br>
</body>
<html:link forward="menu">Presiona para regresar al menÃº</html:link>
</html>
```

Para llamar a esta vista, es necesario cambiar el nombre de la referencia en el ActionServlet AdministrarMenuAction

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

            EmpleadosDepartamentoForm df = new EmpleadosDepartamentoForm();
            DatosDepartamento manejoDpto = new DatosDepartamento();
            LlenarDepartamento llenar = new LlenarDepartamento();
            manejoDpto = llenar.llenarLista();
            df.setDepartamento(manejoDpto);

            request.setAttribute("depto",df);

            return mapping.findForward("listaConTags");



      }
```

Es necesario registrar este forward en las acciones del servlet en el archivo struts-config.xml

```
<action     path="/administrarMenu"     name="ManejoOpcionForm"     scope="request"
type="ts.struts.servlets.AdministrarMenuAction" parameter="opcionElegida">
<forward name="listaConTags" path="/paginas/listaConTags.jsp"></forward>
<forward name="listaDeListas" path="/paginas/listaDeListas.jsp"></forward>
<forward name="listaSencilla" path="/paginas/listaSencilla.jsp"></forward>
</action>
```

Probar la aplicación ejecutando el archivo menu.jsp en el servidor