

Laboratorio. Actualización de datos

Objetivo

Desarrollar una aplicación que muestre los datos almacenados en una base de datos (simulada) en pantalla de tal manera que puedan ser modificados por el usuario para una actualización.

Pasos a seguir.

1. Desarrollar una vista y clases que permitan una validación
2. Desarrollar una vista y clases que desplieguen los datos para una posible modificación

1. Desarrollo de la vista y clases de validación

- Comenzar con un proyecto en blanco y configurar el entorno de desarrollo.
- Crear un workspace llamado ActualizarDatos
- Crear los siguientes paquetes
 - **ts.struts.beans**
 - **ts.struts.modelo**
 - **ts.struts.servlets**
- Crear la carpeta paginas dentro del directorio WebContent
- Cargar el archivo **struts-config.xml** de un proyecto e blanco
- Configurar el archivo **web.xml**

Crear un bean llamado ValidarUsuarioForm en el paquete ts.struts.beans que contendrá los datos a ser leídos en la pantalla de validación.

ValidarUsuarioForm.java

```
package ts.struts.beans;

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;
    }
}
```

Crear un bean llamado DatosUsuarioForm en el paquete **ts.struts.beans** que contendrá los datos a ser desplegados en pantalla para su modificación.

DatosUsuarioForm.java

```
package ts.struts.beans;

import org.apache.struts.action.ActionForm;

public class DatosUsuarioForm extends ActionForm{

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

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

```

Registrar ambos beans en el archivo **struts-config.xml**

```

<form-bean name="DatosUsuarioForm"
type="ts.struts.beans.DatosUsuarioForm"></form-bean>
<form-bean name="ValidarUsuarioForm" type="ts.struts.beans.ValidarUsuarioForm"></
form-bean>

```

Desarrollar la vista para leer los datos de validación

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>Validacion de usuario</title>
</head>
<body>
<h1>Introduce tus datos</h1>

<html:form action="/validarUsuario" method="POST">
Login:
<html:text property="login"></html:text><br><br>
Password:
<html:password property="password"></html:password>
<br><br>
<html:submit>Validar</html:submit>
</html:form>
</body>
</html>

```

Desarrollar el ActionServlet que manejará los datos leídos en la pantalla de validación, éste se encuentra en el paquete **ts.struts.servlets**

ValidarUsuarioAction.java

```
package ts.struts.servlets;

import javax.servlet.http.*;
import org.apache.struts.action.*;
import ts.struts.beans.*;
import ts.struts.modelo.ManejarDatosUsuario;

public class ValidarUsuarioAction extends Action{

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

        DatosUsuarioForm datosUsuario = new DatosUsuarioForm();
        ManejarDatosUsuario manejo = new ManejarDatosUsuario();

        ValidarUsuarioForm vf = (ValidarUsuarioForm)form;
        datosUsuario = manejo.obtenerDatos(vf);
        request.setAttribute("usuario", datosUsuario);

        return mapping.findForward("modificar");

    }

}
```

Este ActionServlet debe ser registrado en el archivo **struts-config.xml** a través de la instrucción

```
<action-mappings>
  <action path="/validarUsuario" name="ValidarUsuarioForm" scope="request"
  type="ts.struts.servlets.ValidarUsuarioAction">
    <forward name="modificar" path="/paginas/modificacion.jsp"></forward>
  </action>
</action-mappings>
```

En este ActionServlet, se obtienen los datos del usuario de la base de datos, los cuales deben ser desplegados en la siguiente vista, por esto es que se le deben pasar a un determinado atributo, en este caso el atributo “usuario” que recibe un objeto tipo ValidarUsusarioForm. Esto es en las instrucciones

```
datosUsuario = manejo.obtenerDatos(vf);
request.setAttribute("usuario", datosUsuario);
```

Desarrollo de la clase que recupere la información de la “base de datos”, esta clase se encuentra en el paquete **ts.struts.modelo**, esta clase también “actualizará” los datos (por el momento solo los desplegará en la Consola). Crear la clase en el paquete **ts.struts.modelo**.

ManejarDatosUsuario.java

```
package ts.struts.modelo;

import ts.struts.beans.*;
public class ManejarDatosUsuario {

    public DatosUsuarioForm obtenerDatos(ValidarUsuarioForm vf){

        DatosUsuarioForm uf = new DatosUsuarioForm();

        uf.setApellidoPaterno("Figueroa");
        uf.setApellidoMaterno("Gonzalez");
        uf.setNombre("Josue");
        uf.setEmail("jfg1978@hotmail.com");
        uf.setTelefono("5529402292");
        uf.setRFC("FIGJ780718");
        uf.setLogin(vf.getLogin());
        uf.setPassword(vf.getPassword());

        return uf;

    }

    public void actualizarDatos(DatosUsuarioForm uf){

        System.out.println("Los nuevos datos son:");
        System.out.println(uf.getApellidoPaterno());
        System.out.println(uf.getApellidoMaterno());
        System.out.println(uf.getNombre());
        System.out.println(uf.getEmail());
        System.out.println(uf.getTelefono());
        System.out.println(uf.getRFC());
        System.out.println(uf.getLogin());
        System.out.println(uf.getPassword());

    }

}
```

2. Desarrollo de la vista para modificación de datos

En el ActionServlet que llama a la vista de despliegue de datos para modificación se le pasó un objeto DatosUsuarioForm a un atributo llamado “usuario” en la vista. Contrario a un despliegue normal, ahora estos datos deben desplegarse en una caja de texto. <html:text>

Recordar agregar las instrucciones para el manejo de tags de html y beans

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

Para desplegar una caja de texto se utiliza la instrucción

```
<html:text property="atributo"></html:text><br><br>
```

En donde “atributo” era el nombre del dato miembro del objeto ActionForm que sería leído en esa caja en particular. Ahora para desplegar la información contenida en ese dato, se debe añadir otro parámetro que contenga ese dato miembro.

```
<html:text name="objeto" property="atributo"></html:text><br><br>
```

En donde “objeto” representa un atributo del tipo ActionForm que se desea desplegar, mientras que “atributo” sigue siendo el dato miembro de ese objeto que se mostrará.

En el ActionServlet que llama a la vista de modificación (ValidarUsuarioAction) se le pasa un objeto tipo DatosUsuarioForm a un atributo llamado “usuario” que se encuentra en la vista modificacion.jsp, esto en la instrucción:

```
request.setAttribute("usuario", datosUsuario);
```

De esta manera la instrucción que despliega uno de los datos miembro (por ejemplo el RFC) del objeto tipo DatosUsuarioForm es:

```
RFC:  
<html:text name="usuario" property="RFC"></html:text><br><br>
```

Así sería posible desplegar los datos que se deseen modificar, el contenido final de la vista es:

modificacion.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>Modificacion de datos</title>  
</head>  
<body>  
<h1>Por favor modifica tus datos:</h1>  
<html:form action="/actualizarDatos" method="POST">  
Nombre:  
<html:text name="usuario" property="nombre"></html:text><br><br>  
Apellido paterno:  
<html:text name="usuario" property="apellidoPaterno"></html:text><br><br>
```

```

Apellido materno:
<html:text name="usuario" property="apellidoMaterno"></html:text><br><br>
Telefono:
<html:text name="usuario" property="telefono"></html:text><br><br>
e-mail:
<html:text name="usuario" property="email"></html:text><br><br>
RFC:
<html:text name="usuario" property="RFC"></html:text><br><br>
<br>
Login:
<html:text name="usuario" property="login"></html:text><br><br>
Password:
<html:text name="usuario" property="password"></html:text><br><br>
<br><br><br><br>
<html:submit>Aceptar</html:submit>
</html:form>
</body>
</html>

```

Esta vista llama al ActionServlet referenciado por la etiqueta “/actualizarDatos” cuya clase se encuentra en el paquete **ts.struts.servlets**, esta clase solicita a la clase ManejarDatosUsuario que “actualice” en la “base de datos”.

Este ActionServlet debe ser registrado en el archivo **struts-config.xml** cuyo contenido final es:

```

<?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="DatosUsuarioForm"
type="ts.struts.beans.DatosUsuarioForm"></form-bean>
    <form-bean name="ValidarUsuarioForm"
type="ts.struts.beans.ValidarUsuarioForm"></form-bean>
  </form-beans>
  <global-exceptions />
  <global-forwards />
  <action-mappings>
    <action path="/validarUsuario" name="ValidarUsuarioForm" scope="request"
type="ts.struts.servlets.ValidarUsuarioAction">
      <forward name="modificar" path="/paginas/modificacion.jsp"></forward>
    </action>
    <action path="/actualizarDatos" name="DatosUsuarioForm" scope="request"
type="ts.struts.servlets.ActualizarDatosAction"></action>
  </action-mappings>
  <controller bufferSize="4096" debug="0" />
  <message-resources parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>

```

ActualizarDatosAction.java

```
package ts.struts.servlets;

import javax.servlet.http.*;

import org.apache.struts.action.*;

import ts.struts.beans.DatosUsuarioForm;
import ts.struts.modelo.ManejarDatosUsuario;

public class ActualizarDatosAction extends Action{

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

        DatosUsuarioForm uf = new DatosUsuarioForm();
        ManejarDatosUsuario manejo = new ManejarDatosUsuario();
        uf=(DatosUsuarioForm)form;
        manejo.actualizarDatos(uf);

        return null;

    }

}
```

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