Laboratorio. Validación de datos

Objetivo.

Utilizar las diferentes propiedades de validación de datos que proporciona el framework Struts.

Actividades

1. Crear un nuevo proyecto
2. Desarrollar una vista para capturar diferentes datos
3. Configurar el entorno de desarrollo para usar validadores
4. Aplicar las diferentes propiedades de validadores.

1. Crear un nuevo proyecto

- Crear un nuevo proyecto
- Configurar el entorno de desarrollo
- Crear los siguientes paquetes
    - **ts.struts.servlets**
    - **ts.struts.beans**

2. Desarrollar una vista para capturar diferentes datos

Crear una forma que permita leer diferentes datos para posteriormente aplicar la propiedad de validación.

En el paquete **ts.struts.beans**, crear la clase **DatosAValidarForm**

DatosAValidarForm.java

```
package ts.struts.beans;

import org.apache.struts.action.ActionForm;

public class DatosAValidarForm extends ActionForm{

	private String nombre;
	private String apellidoPaterno;
	private String apellidoMaterno;
	private String email;
	private String login;
	private String loginRango;
	private String fecha;



	public String getNombre() {
		return nombre;
	}
```

```
	public String getApellidoPaterno() {
		return apellidoPaterno;
	}
	public String getApellidoMaterno() {
		return apellidoMaterno;
	}
	public String getEmail() {
		return email;
	}
	public String getLogin() {
		return login;
	}
	public String getLoginRango() {
		return loginRango;
	}
	public String getFecha() {
		return fecha;
	}
	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 setEmail(String email) {
		this.email = email;
	}
	public void setLogin(String login) {
		this.login = login;
	}
	public void setLoginRango(String loginRango) {
		this.loginRango = loginRango;
	}
	public void setFecha(String fecha) {
		this.fecha = fecha;
	}

}
```

Crear una vista para desplegar cajas de texto que permitan leer estos datos.

leyendoDatos.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>Leyendo datos</title>
<h1>Datos para validar</h1>
<html:form action="/datosLeidos" method="POST"><html:base/>
```

```
Nombre <html:text property="nombre"></html:text><br><br>
Apellido paterno <html:text
property="apellidoPaterno"></html:text><br><br>
Apellido materno <html:text
property="apellidoMaterno"></html:text><br><br>
e-mail <html:text property="email"></html:text><br><br>
Login <html:text property="login"></html:text><br><br>
Login rango <html:text property="loginRango"></html:text><br><br>
Fecha <html:text property="fecha"></html:text><br><br>
<html:submit>Aceptar</html:submit> <html:reset>Limpiar</html:reset>
</html:form>
</head>
<body>
</body>
</html>
```

Crear en el paquete **ts.struts.servlets** la clase DatosLeidosAction que de momento solo imprimirá los datos recibidos.

DatosLeidosAction.java

```
package ts.struts.servlets;

import javax.servlet.http.*;

import org.apache.struts.action.*;

import ts.struts.beans.DatosAValidarForm;

public class DatosLeidosAction extends Action{

      public ActionForward execute(ActionMapping mapping, ActionForm
form,

      HttpServletRequest request,

      HttpServletResponse response){

            DatosAValidarForm vf = new DatosAValidarForm();
            vf = (DatosAValidarForm)form;

            System.out.println(vf.getNombre());
            System.out.println(vf.getApellidoPaterno());
            System.out.println(vf.getApellidoMaterno());
            System.out.println(vf.getLogin());
            System.out.println(vf.getLoginRango());
            System.out.println(vf.getEmail());
            System.out.println(vf.getFecha());


            return null;

      }

}
```

Registrar el servlet y la forma 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="DatosAValidarForm"
type="ts.struts.beans.DatosAValidarForm"></form-bean>
   </form-beans>
   <global-exceptions />
   <global-forwards />
   <action-mappings>
   <action path="/datosLeidos" name="DatosAValidarForm"
scope="request" type="ts.struts.servlets.DatosLeidosAction">
   </action>
   </action-mappings>
   <controller bufferSize="4096" debug="0" />
   <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

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

3. Configurar el entorno de desarrollo para usar validadores

Antes de utilizar los validadores, es necesario realizar modificaciones en el entorno de desarrollo para configurar o instalar los siguientes elementos:

- Plug-in validator
- Archivos de configuración
- La clase ValidatorForm
- Archivo de recursos ApplicationResource.properties

Plug-in validator

El Plug-in validator es una clase encargada del manejo de las validaciones automáticas dentro de una aplicación web. Para que estas validaciones es necesario registrar esta clase en el archivo **struts-config.xml** agregando las siguientes líneas:

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
<set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
```

Archivos de configuración

Es necesario agregar dos archives adicionales al entorno de desarrollo: validator-rules.xml y validation.xml.

Archivo validator-rules.xml

Este archivo contiene la declaración de clases de validación y los métodos utilizados para realizar las diferentes rutinas de validación predefinidas.

Archivo validation.xml

En este archivo, se debe asignar a cada campo que se quiera validar la o las reglas de validación definidas en el archivo validator-rules.xml.

Para cada formulario que se desee validar se debe definir un elemento <form> cuyo atributo *name* deberá contener el nombre del objeto ActionForm encargado de capturar el dato del usuario. Cada campo que se quiera validar del formulario se define en el interior de<form> a través del elemento <field>.

Clase ValidatorForm

Es una clase que se encuentra definida en el paquete **org.apache.struts.validator** y es una sub clase de ActionForm. En ella se invocan los diferentes métodos de validación definidosen el archivo validator-rules.xml, según la información suministrada en el archivo validation.xml.

Archivo ApplicationResource.properties

Almacena cadenas de texto que pueden tener diferentes usos dentro de la aplicación, en el caso de los validadores, se almacenan los mensajes de error devueltos por cada validador cuando no se cumpla el criterio de validación definido por los campos. En el archivo validator-rules.xml vienen comentados los mensajes de error por defecto en struts.

```
errors.required={0} is required.
errors.minlength={0} can not be less than {1} characters.
errors.maxlength={0} can not be greater than {1} characters.
errors.invalid={0} is invalid.

errors.byte={0} must be a byte.
errors.short={0} must be a short.
errors.integer={0} must be an integer.
errors.long={0} must be a long.
errors.float={0} must be a float.
errors.double={0} must be a double.

errors.date={0} is not a date.
errors.range={0} is not in the range {1} through {2}.
errors.creditcard={0} is an invalid credit card number.
errors.email={0} is an invalid e-mail address.
```

Instalación y configuración de archivos.

Plug-in validator

Para instalar el plugin, es necesario agregar las líneas antes mencionadas 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="DatosAValidarForm"
type="ts.struts.beans.DatosAValidarForm"></form-bean>
   </form-beans>
   <global-exceptions />
   <global-forwards />
   <action-mappings>
   <action path="/datosLeidos" name="DatosAValidarForm"
scope="request" type="ts.struts.servlets.DatosLeidosAction">
   </action>
   </action-mappings>
   <controller bufferSize="4096" debug="0" />
   <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
   <set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
   </plug-in>
```

```
</struts-config>
```

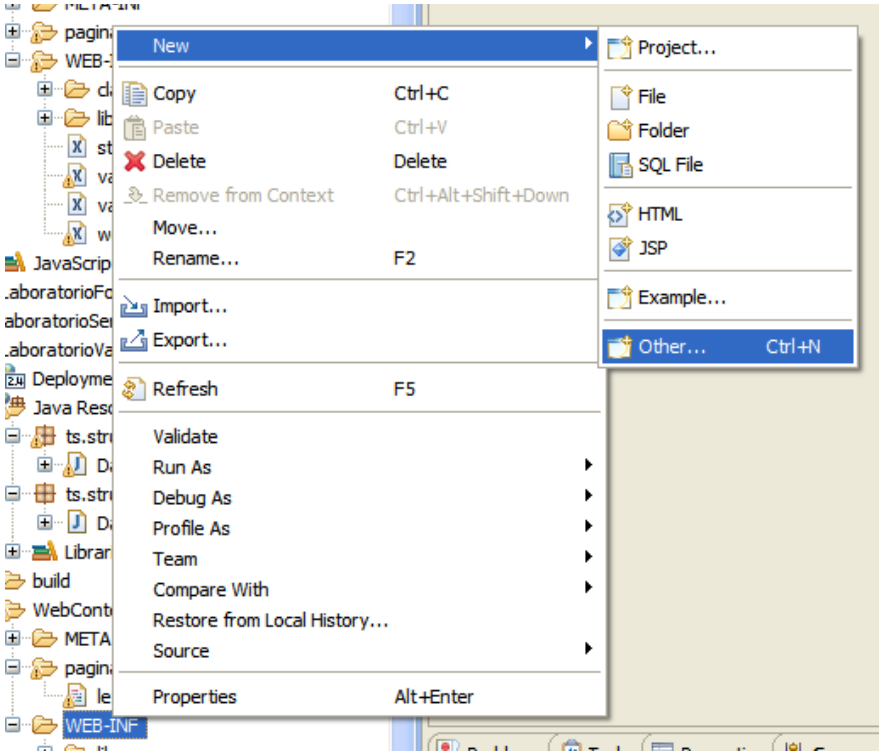
Archivo validator-rules.xml

Descargar el archivo.

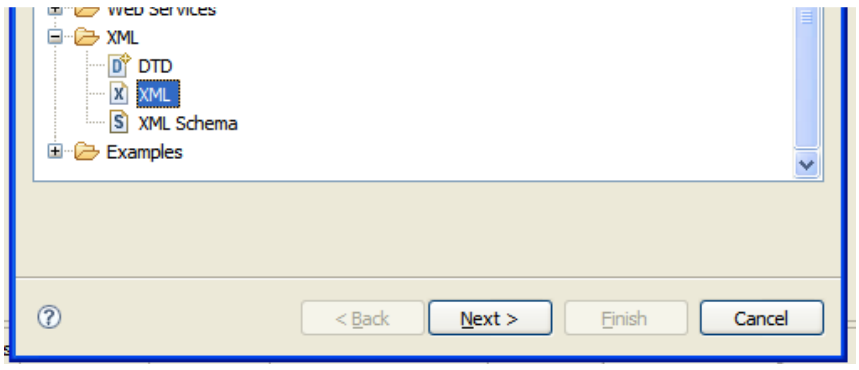
Archivo validation.xml

Este archivo se debe crear por lo que inicialmente estaría vacío.

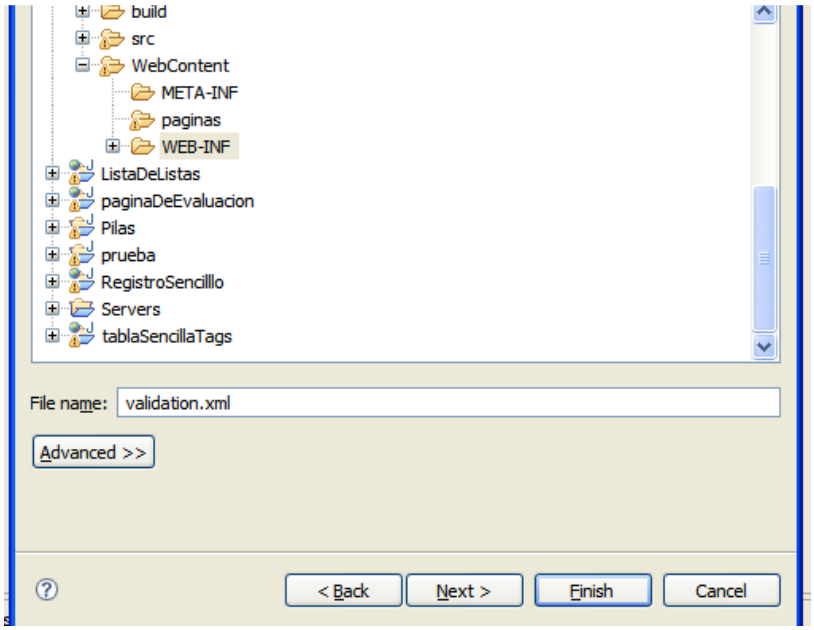
- Clic derecho sobre la carpeta WEB-INF
- Clic New -> Other



- Seleccionar el ícono de folder XML y posteriormente XML

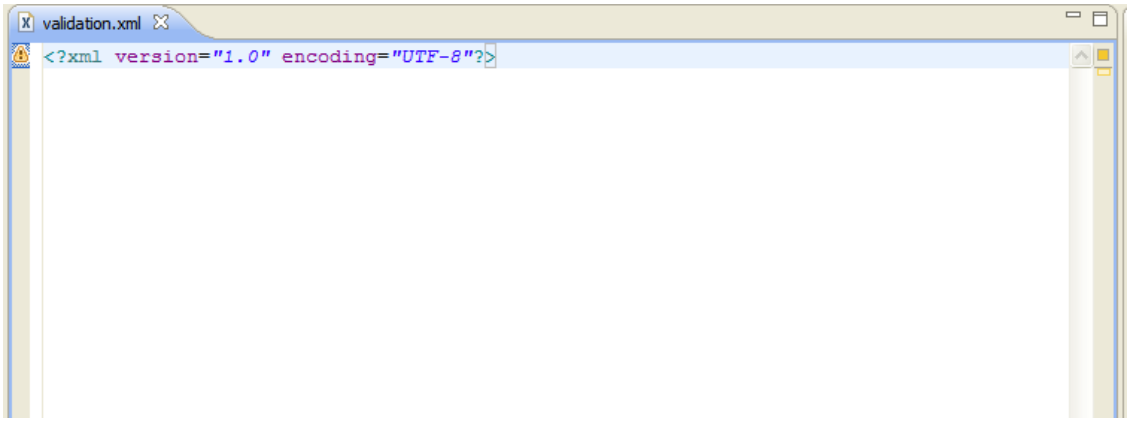


- Clic Next
- Escribir en el campo de nombre **validation.xml**

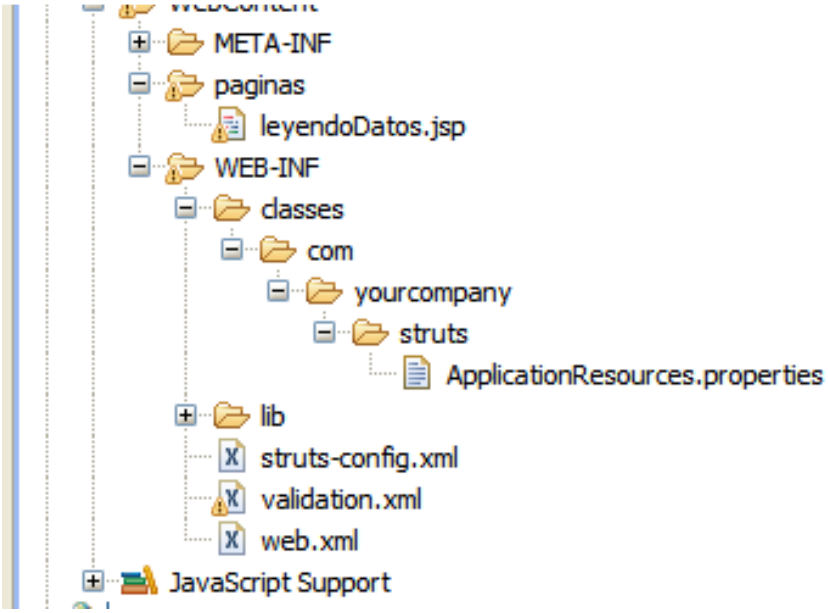


- Clic Finish

Se creará el archivo validation.xml vacío



Del proyecto en blanco de Struts, se debe copiar la carpeta clases de la carpeta WEB-INF a la carpeta WEB-INF del workspace del proyecto. En esta carpeta se encuentra el archivo **ApplicationResource.properties.**



Por el momento, el archivo ApplicationResources.properties está vacío, se copiarán los errores contenidos en el archivo validation-rules.xml a este archivo.

ApplicationResources.properties

```
# Resources for parameter
'com.yourcompany.struts.ApplicationResources'
# Project P/StrutsBlank


   errors.required={0} is required.
   errors.minlength={0} can not be less than {1} characters.
   errors.maxlength={0} can not be greater than {1} characters.
   errors.invalid={0} is invalid.

   errors.byte={0} must be a byte.
   errors.short={0} must be a short.
   errors.integer={0} must be an integer.
   errors.long={0} must be a long.
   errors.float={0} must be a float.
   errors.double={0} must be a double.

   errors.date={0} is not a date.
   errors.range={0} is not in the range {1} through {2}.
   errors.creditcard={0} is an invalid credit card number.
   errors.email={0} is an invalid e-mail address.
```

Para poder realizar una validación en el navegador, es necesario incluir una instrucción del tipo:

```
<html:javascript formName = “objeto_Form”/>
```

En donde objeto_Form es el nombre del Form del que se desean validar atributos.

También se debe incluir el evento *onsubmit* en donde se declara la acción y el método a utilizar, el nombre que se debe colocar es *validateObjetoForm*, en donde ObjetoForm es el nombre del Form del que se desean validar campos. La llamada a este método debe incluir el argumento *this*.

```
onsubmit=”return validateObjetoForm(this);”
```

4. Aplicar las propiedades de validadores

Lo primero es cambiar la el Form del que se desean validar los campos para que herede de la clase ValidatorForm en lugar de ActionForm

DatosAValidarForm.java

```
import org.apache.struts.action.ActionForm;
import org.apache.struts.validator.ValidatorForm;
public class DatosAValidarForm extends ValidatorForm{
	private String nombre;
	private String apellidoPaterno;
	private String apellidoMaterno;
	private String email;
	private String login;
	private String loginRango;
	private String fecha;
	public String getNombre() {
		return nombre;
	}
	public String getApellidoPaterno() {
		return apellidoPaterno;
	}
	public String getApellidoMaterno() {
		return apellidoMaterno;
	}
	public String getEmail() {
		return email;
	}
	public String getLogin() {
		return login;
	}
	public String getLoginRango() {
		return loginRango;
	}
	public String getFecha() {
		return fecha;
	}
	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 setEmail(String email) {
		this.email = email;
	}
	public void setLogin(String login) {
		this.login = login;
	}
	public void setLoginRango(String loginRango) {
		this.loginRango = loginRango;
	}
	public void setFecha(String fecha) {
		this.fecha = fecha;
	}
}
```

Configurando el archivo validation.xml

Lo primero es agregar las siguientes líneas para poder declarar dentro de ellas las validaciones deseadas.

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




</formset>
</form-validation>
```

De momento se validará que se haya escrito el nombre a registrar, para esto se agrega la siguiente información en el archivo validation.xml

```
<form name="DatosAValidarForm">
<field property="nombre" depends="required">
<arg key="DatosAValidarForm.nombre"/>
</field>
</form>
```

Es necesario registrar la información en el archivo ApplicationResources.properties agregando la siguiente información:

```
DatosAValidarForm.nombre=nombre
```

Modificaciones a la vista.

leyendoDatos.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>Leyendo datos</title>
<html:javascript formName="DatosAValidarForm"/>
<h1>Datos para validar</h1>
<html:form action="/datosLeidos" method="POST" onsubmit="return
validateDatosAValidarForm(this);"><html:base/>
Nombre <html:text property="nombre"></html:text><br><br>
Apellido paterno <html:text
property="apellidoPaterno"></html:text><br><br>
Apellido materno <html:text
property="apellidoMaterno"></html:text><br><br>
e-mail <html:text property="email"></html:text><br><br>
Login <html:text property="login"></html:text><br><br>
```

```
Login rango <html:text property="loginRango"></html:text><br><br>
Fecha <html:text property="fecha"></html:text><br><br>
<html:submit>Aceptar</html:submit> <html:reset>Limpiar</html:reset>
</html:form>
</head>
<body>
</body>
</html>
```

Al probar la aplicación, si no se introduce un valor en el campo nombre, se presenta la siguiente pantalla.

De esta manera, se comenzará por hacer necesarios todos los campos:

ApplicationResources.properties

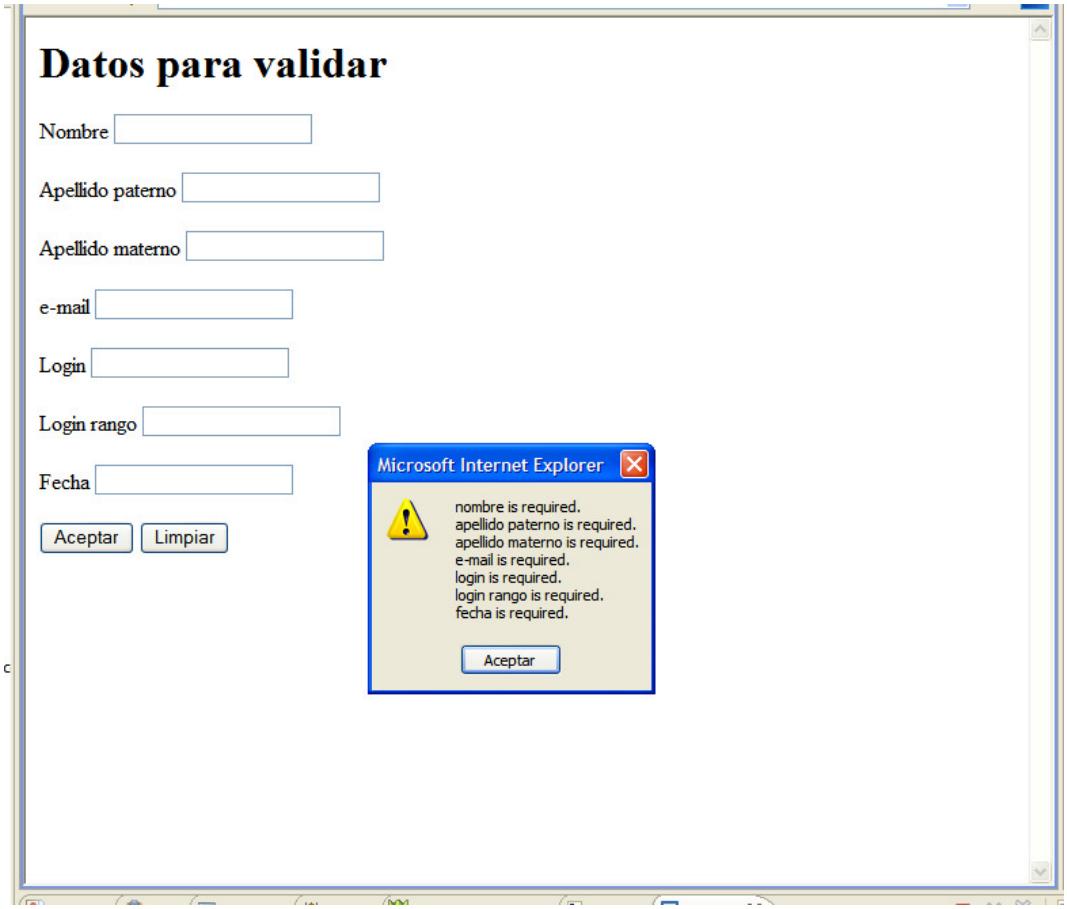
```
# Resources for parameter
'com.yourcompany.struts.ApplicationResources'
# Project P/StrutsBlank

DatosAValidarForm.nombre=nombre
DatosAValidarForm.apellidoPaterno=apellido paterno
DatosAValidarForm.apellidoMaterno=apellido materno
DatosAValidarForm.email=e-mail
DatosAValidarForm.login=login
DatosAValidarForm.loginRango=login rango
DatosAValidarForm.fecha=fecha

   errors.required={0} is required.
   errors.minlength={0} can not be less than {1} characters.
   errors.maxlength={0} can not be greater than {1} characters.
   errors.invalid={0} is invalid.
   errors.byte={0} must be a byte.
   errors.short={0} must be a short.
   errors.integer={0} must be an integer.
   errors.long={0} must be a long.
   errors.float={0} must be a float.
   errors.double={0} must be a double.
   errors.date={0} is not a date.
   errors.range={0} is not in the range {1} through {2}.
   errors.creditcard={0} is an invalid credit card number.
   errors.email={0} is an invalid e-mail address.
```

validation.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<form-validation>
<formset>
<form name="DatosAValidarForm">
<field property="nombre" depends="required">
<arg key="DatosAValidarForm.nombre"/>
</field>
<field property="apellidoPaterno" depends="required">
<arg key="DatosAValidarForm.apellidoPaterno"/>
</field>
<field property="apellidoMaterno" depends="required">
<arg key="DatosAValidarForm.apellidoMaterno"/>
</field>
<field property="email" depends="required">
<arg key="DatosAValidarForm.email"/>
</field>
<field property="login" depends="required">
<arg key="DatosAValidarForm.login"/>
</field>
<field property="loginRango" depends="required">
<arg key="DatosAValidarForm.loginRango"/>
</field>
<field property="fecha" depends="required">
<arg key="DatosAValidarForm.fecha"/>
</field>
</form>
```

```
</formset>
</form-validation>
```

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



Otros tipos de validaciones

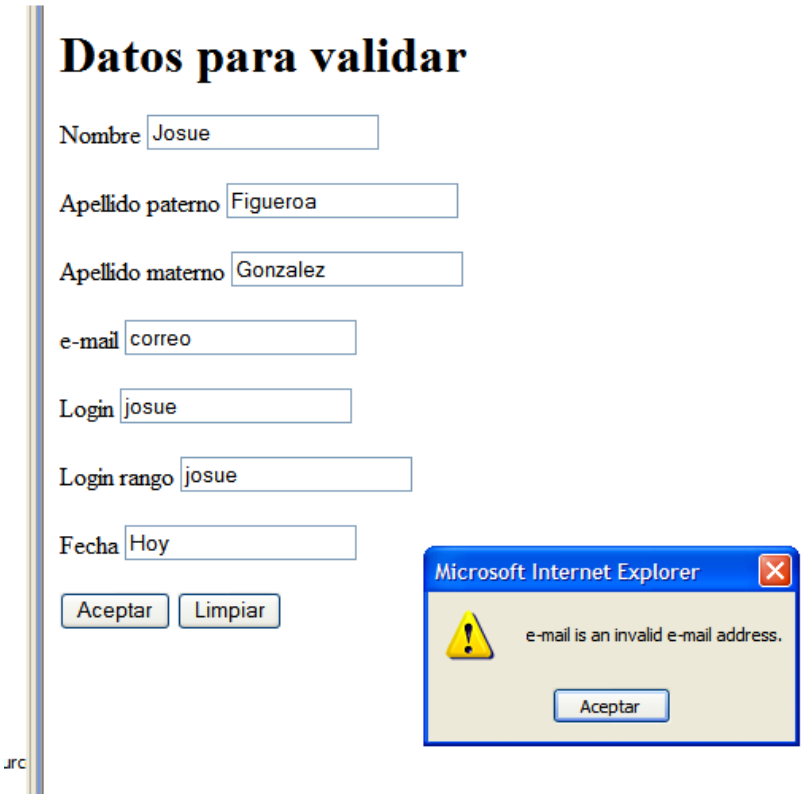
Como se observa, existen otros tipos de validaciones a demás de "required", algunas relacionadas con los formatos de los datos, como las fechas y los correos electrónicos.

Validando e-mails

Este validador ya está definido, por lo que para usarlo solo basta agregar su parámetro en el atributo *depends*, de esta manera es posible combinar que el correo sea necesario y adicionalmente tenga un formato de correo electrónico. Esto en el archivo validation.xml.

```
<field property="email" depends="required,email">
<arg key="DatosAValidarForm.email"/>
</field>
```

Probando la aplicación, llenando todos los campos, pero sin usar un formato de correo electrónico correcto se tiene:



Validando fechas

Otro campo que puede requerir un formato específico es una fecha, para esto se agrega el valor *date* en el atributo *depends.* Es posible, através de las etiquetas <var>, <var-name> y <var-value> el formato para la fecha.

```
<var>
<var-name>datePattern>/var-name>
<var-value>dd/MM/yyyy</var-value>
```

De esta manera, para validar un format de fecha, se realizan las siguientes modificaciones en el archivo validation.xml.

```
</field>
<field property="fecha" depends="required,date">
<arg key="DatosAValidarForm.fecha"/>
<var>
<var-name>datePattern</var-name>
<var-value>dd/MM/yyyy</var-value>
</var>
</field>
```

Probando la aplicación:

Estableciendo rangos de caracteres

En algunas ocasiones se desea limitar la cantidad de caracteres a un mínimo y un máximo (por ejemplo para leer un login y un password).

Para esto es necesario utilizar en principio, el atributo *minlength*:

En el archivo **validation.xml**

```
<field property="loginRango" depends="required,minlength">
<arg0 key="DatosAValidarForm.loginRango"/>
<arg1 name="minlength" key="DatosAValidarForm.loginRango.min"/>
<var>
<var-name>minlength</var-name>
<var-value>5</var-value>
</var>
</field>
```

Se coloca arg1 por que el error de *minlenght* necesita dos argumentos, {0} y {1}

En el archivo **ApplicationResources.properties**

```
DatosAValidarForm.loginRango=login rango
DatosAValidarForm.loginRango.min="5"
```

Estableciendo un límite máximo:

El procedimiento es similar al límite máximo

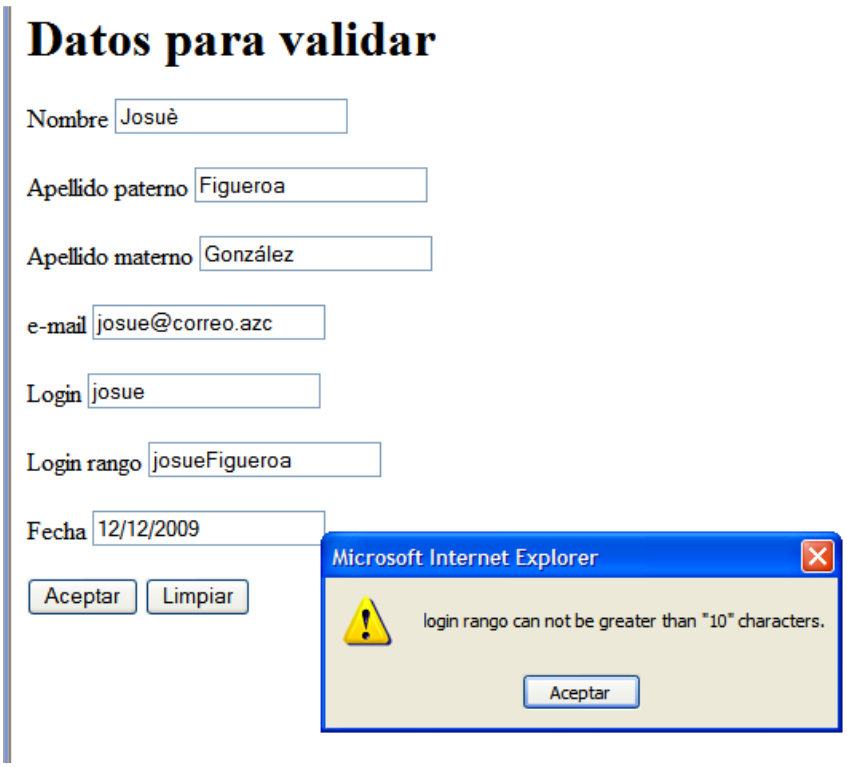
En el archivo **validation.xml**

```
<field property="loginRango" depends="required,minlength,maxlength">
<arg0 key="DatosAValidarForm.loginRango"/>
<arg1 name="minlength" key="DatosAValidarForm.loginRango.min"/>
<arg1 name="maxlength" key="DatosAValidarForm.loginRango.max"/>
<var>
<var-name>minlength</var-name>
<var-value>5</var-value>
<var-name>maxlength</var-name>
<var-value>10</var-value>
</var>
</field>
```

Se coloca arg1 por que el error de *maxlenght* necesita dos argumentos, {0} y {1}

En el archivo **ApplicationResources.properties**

```
DatosAValidarForm.loginRango=login rango
DatosAValidarForm.loginRango.min="5"
DatosAValidarForm.loginRango.max="10"
```



Personalizando mensajes de errores

La forma más sencilla de personalizar errores es modificar los mensajes de los errores ya predefinidos en el *framework* Struts.

Si bien se pueden modificar los mensajes ya predefinidos en Struts, lo más conveniente es crear nuevos mensajes personalizados. Para esto es necesario:

Añadir el mensaje al archivo de recursos. Agregar el nuevo mensaje al archivo ApplicationResources.properties.

Indicar el mensaje a utilizar en el campo <field>. Se debe utilizar el elemento <msg> de <field> indicando a través de sus atributos *name* y *key* el nombre del validador al que se asocia el mensaje.

Ejemplo:

Se creará un mensaje personalizado para cuando se verifican las propiedades de email (requerido y formato especial).

En este caso se hará referencia a los nuevos mensajes a través de las siguientes referencias:

errores.personalizados.email.requerido – Para indicar que es un campo necesario

errores.personalizados.email.formato – Para indicar que es un formato de e-mail

En el archivo **ApplicationResources.properties** se agregan las referencias y los mensajes que se desean desplegar.

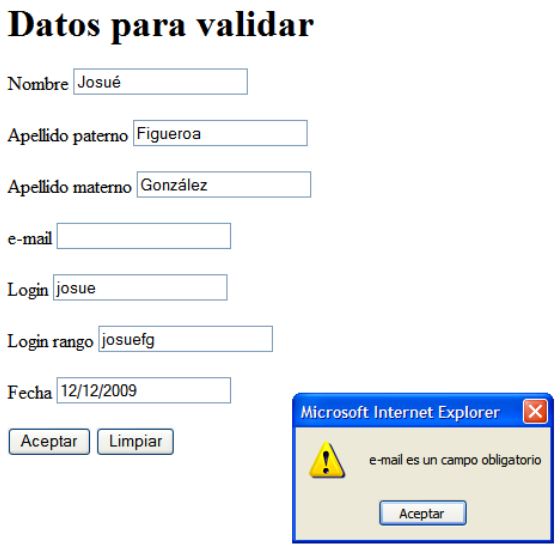
```
errores.personalizados.email.requerido = {0} es un campo obligatorio
errores.personalizados.email.formato = {0} no es un formato de correo
electrónico válido
```

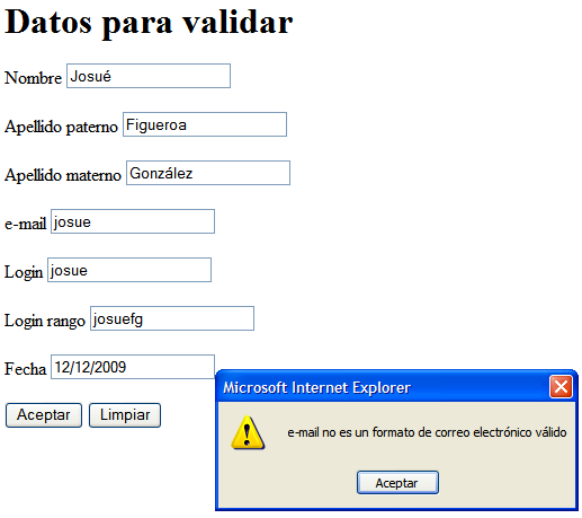
En el archivo **validation.xml** se debe indicar en el campo de validación del email que se utilizarán mensajes personalizados.

```
<field property="email" depends="required,email">
<msg name="email" key="errores.personalizados.email.formato"/>
<msg name="required" key="errores.personalizados.email.requerido"/>
<arg0 key="DatosAValidarForm.email"/>
</field>
```

En este caso, en el atributo *name* se mantiene el tipo de validador a utilizar y en el campo *key* se coloca la referencia del mensaje que se desea desplegar.

Probando la aplicación:





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

Personalizando validadores

Una de las formas de crear nuevos validadores es sobre escribir el método valídate que está disponible en la clase ValidatorForm la cuál es extendida por la forma en la que se desean validar campos.

En este caso se desean validar los elementos de la forma ComparaPasswordsForm, por lo que se debe sobre escribir el método valídate() para que realice esta validación.

La firma del método valídate() es la siguiente:

```
public ActionErrors valídate(ActionMapping mapping, HttpServletRequest request)
```

ComparaPasswordsForm.java

```
package ts.struts.beans;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.validator.ValidatorForm;

public class ComparaPasswordsForm extends ValidatorForm{

	private String passOriginal;
	private String passRepetido;

	public String getPassOriginal() {
		return passOriginal;
	}
	public String getPassRepetido() {
		return passRepetido;
	}
	public void setPassOriginal(String passOriginal) {
		this.passOriginal = passOriginal;
	}
	public void setPassRepetido(String passRepetido) {
		this.passRepetido = passRepetido;
	}

	public ActionErrors validate(ActionMapping mapping,

	HttpServletRequest request){

		ActionErrors errores = super.validate(mapping, request);

		String passwordOriginal = getPassOriginal();
		String passwordRepetido = getPassRepetido();

		if(passOriginal.contains(".")|| passOriginal.contains("-")
				|| passOriginal.contains("_")||
passOriginal.contains(","))
```

```
                    errores.add("passOriginal", new
ActionMessage("errores.personalizados.password.noValido"));

            return errores;

        }

}
```

Es necesario agregar dos atributos más al mapeo de la acción 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="DatosAValidarForm"
type="ts.struts.beans.DatosAValidarForm"></form-bean>
   <form-bean name="ComparaPasswordsForm"
type="ts.struts.beans.ComparaPasswordsForm"></form-bean>
   </form-beans>
   <global-exceptions />
   <global-forwards />
   <action-mappings>
   <action path="/datosLeidos" name="DatosAValidarForm"
scope="request" type="ts.struts.servlets.DatosLeidosAction">
   </action>
   <action path="/comparaPasswords" name="ComparaPasswordsForm"
scope="request" type="ts.struts.servlets.ComparaPasswordsAction"
      validate="true" input="/paginas/validaPassword.jsp">
   </action>
   </action-mappings>
   <controller bufferSize="4096" debug="0" />
   <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
   <set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
   </plug-in>
</struts-config>
```

En la vista es necesario agregar las instrucciones para el uso de validadores, y adicionalmente la instrucción

```
<html:errors property="nombre"/>
```

En donde *nombre* representa el atributo que tiene asociado un posible error.

validaPassword.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>Validando passwords</title>
<html:javascript formName="ComparaPasswordsForm"/>
</head>
<body>
<h1>Introduce un password</h1>
Los passwords introducidos deben ser iguales, además de no contener
ninguno de los siguientes caracteres,
coma (,) guiones (- _) o puntos (: .)<br><br>

<html:form action="/comparaPasswords" method="POST" onsubmit="return
validateComparaPasswordsForm(this);">

Password: <html:password
property="passOriginal"></html:password><br><br>
<html:errors property="passOriginal"/><br><br>

Repite el password: <html:password
property="passRepetido"></html:password><br><br>
<html:submit>Aceptar</html:submit>

</html:form>
</body>
</html>
```

Finalmente se agrega el mensaje a desplegar en el archivo ApplicationResources.properties

```
errores.personalizados.password.noValido = El password no puede
contener (, . - _)
```

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

Se introduce como password **josue.figueroa**

De manera similar se coloca la condición de que los passwords deben ser iguales.

De esta manera los archivos quedan:

validaPassword.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>Validando passwords</title>
<html:javascript formName="ComparaPasswordsForm"/>
</head>
<body>
<h1>Introduce un password</h1>
Los passwords introducidos deben ser iguales, además de no contener
ninguno de los siguientes caracteres,
coma (,) guiones (- _) o puntos (: .)<br><br>

<html:form action="/comparaPasswords" method="POST" onsubmit="return
validateComparaPasswordsForm(this);">

Password: <html:password
property="passOriginal"></html:password><br><br>
<html:errors property="passOriginal"/><br><br>

Repite el password: <html:password
property="passRepetido"></html:password><br><br>
<html:errors property="passRepetido"/><br><br>

<html:submit>Aceptar</html:submit>

</html:form>
</body>
</html>
```

ComparaPasswordsForm.java

```
package ts.struts.beans;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.validator.ValidatorForm;

public class ComparaPasswordsForm extends ValidatorForm{

	private String passOriginal;
	private String passRepetido;

	public String getPassOriginal() {
		return passOriginal;
	}
	public String getPassRepetido() {
		return passRepetido;
	}
	public void setPassOriginal(String passOriginal) {
		this.passOriginal = passOriginal;
	}
	public void setPassRepetido(String passRepetido) {
		this.passRepetido = passRepetido;
	}

	public ActionErrors validate(ActionMapping mapping,
```

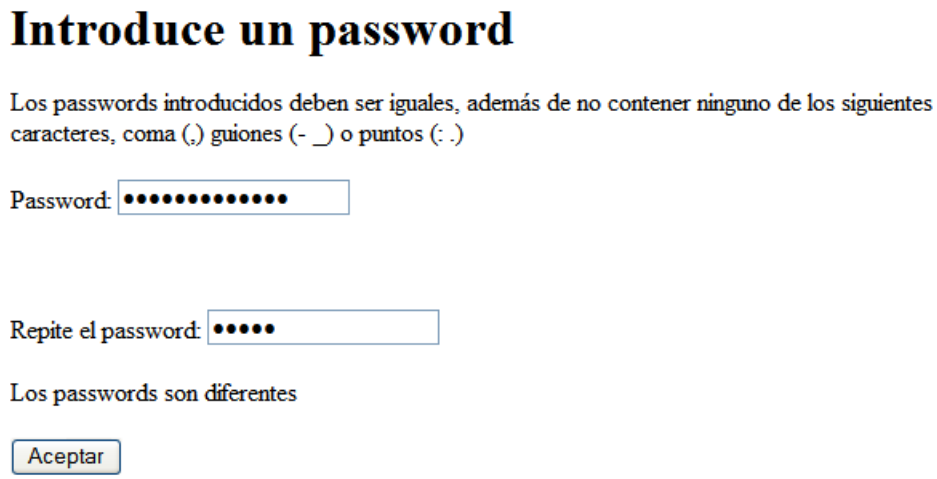
```
      HttpServletRequest request){

            ActionErrors errores = super.validate(mapping, request);

            String passwordOriginal = getPassOriginal();
            String passwordRepetido = getPassRepetido();

            if(passOriginal.contains(".")|| passOriginal.contains("-")
                        || passOriginal.contains("_")||
passOriginal.contains(",")){
                  errores.add("passOriginal", new
ActionMessage("errores.personalizados.password.noValido"));
                  return errores;
            }

            else if(passOriginal.compareTo(passRepetido)!=0){
                  errores.add("passRepetido", new
ActionMessage("errores.personalizados.password.noIguales"));
                  return errores;
            }
            return null;

      }

}
```

Al archivo **ApplicationResources.properties** se agrega la siguiente línea

```
errores.personalizados.password.noIguales = Los passwords son
diferentes
```

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

Probando la aplicación introduciendo dos passwords diferentes



Combinando validadores

Es posible combinar el uso de validadores predefinidos por Struts con los personalizados.

En este caso es necesario que los passwords no se envíen en blanco, por lo que se utilizará el validador *required* ya predefinido en Struts.

La forma de utilizarla no cambia respecto a los manejos anteriores, por lo que los archivos quedan de la siguiente manera:

validation.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<form-validation>
<formset>
<form name="DatosAValidarForm">
<field property="nombre" depends="required">
<arg key="DatosAValidarForm.nombre"/>
</field>
<field property="apellidoPaterno" depends="required">
<arg key="DatosAValidarForm.apellidoPaterno"/>
</field>
<field property="apellidoMaterno" depends="required">
<arg key="DatosAValidarForm.apellidoMaterno"/>
</field>
<field property="email" depends="required,email">
<msg name="email" key="errores.personalizados.email.formato"/>
<msg name="required" key="errores.personalizados.email.requerido"/>
<arg0 key="DatosAValidarForm.email"/>
</field>
<field property="login" depends="required">
<arg key="DatosAValidarForm.login"/>
</field>
<field property="loginRango" depends="required,minlength,maxlength">
<arg0 key="DatosAValidarForm.loginRango"/>
<arg1 name="minlength" key="DatosAValidarForm.loginRango.min"/>
<arg1 name="maxlength" key="DatosAValidarForm.loginRango.max"/>
<var>
<var-name>minlength</var-name>
<var-value>5</var-value>
<var-name>maxlength</var-name>
<var-value>10</var-value>
</var>
</field>
<field property="fecha" depends="required,date">
<arg key="DatosAValidarForm.fecha"/>
<var>
<var-name>datePattern</var-name>
<var-value>dd/MM/yyyy</var-value>
</var>
</field>
</form>
<form name="ComparaPasswordsForm">
<field property="passOriginal" depends="required">
<arg key="ComparaPasswordsForm.passOriginal"/>
</field>
<field property="passRepetido" depends="required">
<arg key="ComparaPasswordsForm.passRepetido"/>
</field>
```

```
</form>
</formset>
</form-validation>
```

ApplicationResources.properties

```
# Resources for parameter
'com.yourcompany.struts.ApplicationResources'
# Project P/StrutsBlank


DatosAValidarForm.nombre=nombre
DatosAValidarForm.apellidoPaterno=apellido paterno
DatosAValidarForm.apellidoMaterno=apellido materno
DatosAValidarForm.email=e-mail
DatosAValidarForm.login=login
DatosAValidarForm.loginRango=login rango
DatosAValidarForm.loginRango.min="5"
DatosAValidarForm.loginRango.max="10"
DatosAValidarForm.fecha=fecha

ComparaPasswordsForm.passOriginal=el password
ComparaPasswordsForm.passRepetido=el password repetido

   errores.personalizados.email.requerido = {0} es un campo
obligatorio
   errores.personalizados.email.formato = {0} no es un formato de
correo electrónico válido
   errores.personalizados.password.noValido = El password no puede
contener (, . - _)
   errores.personalizados.password.noIguales = Los passwords son
diferentes

   errors.required={0} is required.
   errors.minlength={0} can not be less than {1} characters.
   errors.maxlength={0} can not be greater than {1} characters.
   errors.invalid={0} is invalid.

   errors.byte={0} must be a byte.
   errors.short={0} must be a short.
   errors.integer={0} must be an integer.
   errors.long={0} must be a long.
   errors.float={0} must be a float.
   errors.double={0} must be a double.

   errors.date={0} is not a date.
   errors.range={0} is not in the range {1} through {2}.
   errors.creditcard={0} is an invalid credit card number.
   errors.email={0} is an invalid e-mail address.
```

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

Introduce un password

Los passwords introducidos deben ser iguales, además de no contener ninguno de los siguientes caracteres, coma (,) guiones (- _) o puntos (: .)

Password:

Repite el password:

Aceptar