



Struts

**TAGS STRUTS**

# TAGS STRUTS

- Tags de Struts:

- Struts 2.x:

- <http://struts.apache.org/release/2.0.x/docs/tag-reference.html>

- Tutorial más detallado que en los apuntes:

- <http://www.javatutoriales.com/2013/11/struts-2-parte-7-tags.html>

# TAGS Struts

Permiten la creación de aplicaciones RIA con mínimo código.

Pueden dividirse en dos grupos:

- Etiquetas Genéricas de Struts
  - Etiquetas de Control
  - Etiquetas de Datos
- Etiquetas de interfaz de usuario
  - Etiquetas de Formulario
  - Etiquetas de No-Formulario

# Etiquetas Genericas de Struts

---

Etiquetas de  
control y de  
manejo de  
datos

# TAGS de CONTROL

- Tags de Struts. **Control tags:**

- Conjunto de etiquetas de control de ejecución:

- if

- elseif

- else

```
<s:if test="%{false}">
    <div>Will Not Be Executed</div>
</s:if>
<s:elseif test="%{true}">
    <div>Will Be Executed</div>
</s:elseif>
<s:else>
    <div>Will Not Be Executed</div>
</s:else>
```

# TAGS de CONTROL

- **Control tags:**
  - **append.** Concatena objeto de tipo iterator (ArrayList).
  - **generator.** Genera un iterator (objeto iterable).
  - **iterator.** Itera sobre un objeto de tipo `java.util.Collection` o `java.util.Iterator`.
  - **merge.** Une objetos de tipo iterator intercalando sus componentes.
  - **sort.** Ordena una lista utilizando un `Comparator`.
  - **subset.** Obtiene un subconjunto de una lista.

# TAGS de DATOS

- **Data tags:**

- Conjunto de etiquetas de manejo de datos.

- **a.** Etiqueta `<a>` de HTML.

- `<s:a href="%{url}">English</s:a>`

- **action.** Permite invocar a un Action directamente desde un JSP.

- **bean.** Instancia un Bean.

- **date.** Permite dar formato a un objeto Date.

- **debug.** Muestra el contenido de la pila(valueStack).

- **include.** Obtiene un recurso y lo hace accesible.

- **include.** Incluye un jsp o un servlet.

# TAGS de DATOS

- **Data tags:**

- **param.** Indica un parámetro a pasar (por ejemplo a un include).
- **property.** Muestra el valor de una propiedad.
- **push.** Coloca un valor en la cima del ValueStack.
- **set.** Asigna valor a una variable.
- **text.** Muestra un texto i18n.
- **url.** Crea una URL.

```
<s:url id="url" action="HelloWorld">
    <s:param name="request_locale">en</s:param>
</s:url>
<s:a href="%{url}">English</s:a>
```

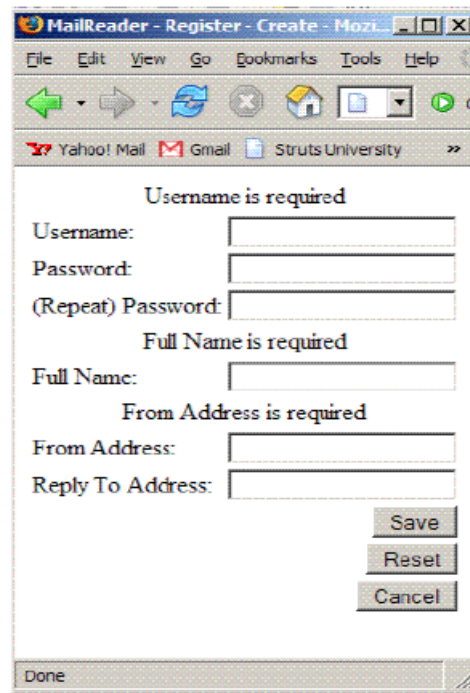


# Etiquetas UI

---

Etiquetas de  
Formulario y  
otras

# Etiquetas de formulario



The image shows a screenshot of a web browser window titled "MailReader - Register - Create - Mozilla". The browser's address bar displays "Yahoo! Mail", "Gmail", and "Struts University". The main content area contains a registration form with the following elements:

- Username is required**: A label above a text input field.
- Password:**: A label above a text input field.
- (Repeat) Password:**: A label above a text input field.
- Full Name is required**: A label above a text input field.
- From Address is required**: A label above a text input field.
- Reply To Address:**: A label above a text input field.
- Buttons**: Three buttons labeled "Save", "Reset", and "Cancel" are positioned at the bottom right of the form.
- Done**: A button located at the bottom left of the browser window.

# Etiquetas de Formularios

Sin Struts (Solo una parte del formulario)

```
<% User user = ActionContext.getContext() %>
<form action="Profile_update.action" method="post">
<table>
<tr>
<td align="right"><label>First name:</label></td>
<td><input type="text" name="user.firstname"
value="<%=user.getFirstname() %> /></td>
</tr>
<tr>
<td>
<input type="radio" name="user.gender" value="0"
id="user.gender0"
<% if (user.getGender()==0) { %>
checked="checked" %> } %> />
<label for="user.gender0">Female</label>
```

...

# Etiquetas de Formularios

con Struts (Formulario completo)

```
<s:actionerror/>
<s:form action="Profile_update" validate="true">
  <s:textfield label="Username" name="username"/>
  <s:password label="Password" name="password"/>
  <s:password label="(Repeat) Password" name="password2"/>
  <s:textfield label="Full Name" name="fullName"/>
  <s:textfield label="From Address" name="fromAddress"/>
  <s:textfield label="Reply To Address"
name="replyToAddress"/>
  <s:submit value="Save" name="Save"/>
  <s:submit action="Register_cancel" value="Cancel"
name="Cancel" onclick="form.onsubmit=null"/>
</s:form>
```

# Etiquetas de Formularios

- Tags de Struts. **Form tags:**

- checkbox**. Muestra un elemento HTML de tipo checkbox.
- checkboxlist**. Crea una serie de checkbox a partir de un objeto List o Map.
- combobox**. Crea una lista desplegable y una caja de texto y los vincula.
- doubleselect**. Crea dos listas desplegables vinculadas.
- head**. Agrega elementos (css, js) en la sección “head” de la página HTML.
- file**. Muestra un componente para la subida de ficheros.
- form**. Muestra un formulario HTML.
- hidden**. Crea una campo de tipo hidden.
- label**. Muestra una etiqueta “label” de HTML.

# Etiquetas de Formularios

- Tags de Struts. **Form tags:**

- optiontransferselect.** Componente de asignación de una lista a otra.
- optgroup.** Crea una etiqueta optgroup dentro de una etiqueta select.
- password.** Crea una etiqueta input de tipo password.
- radio.** Crea una elemento HTML de tipo radio.
- reset.** Crea un componente de tipo reset.
- select.** Crea una etiqueta de tipo select y los option correspondientes.
- submit.** Crea un botón de tipo submit.
- textarea.** Crea un componente textarea de HTML.
- textfield.** Crea un componente text de HTML.
- token.** Detiene el doble submit de un formulario.
- updownselect.** Crea una lista de selección con botones para ordenar los elementos.

# Etiquetas No-Formularios

- **Non-Form UI tags:**

- Muestran información de lo ocurrido en los Action.
- **actionerror**. Muestra un mensaje de error generado por un Action.
- **actionmessage**. Muestra un mensaje generado por un Action.
- **component**. Muestra un componente propio cuando se usa un template.
- **div**. Crea una etiqueta div de HTML. Útil cuando se utiliza el Ajax Theme.
- **fielderror**. Muestra un error relacionado con un campo de un formulario.

# Struts

- Tags de Struts. **Ajax tags:**

- a
- autocompleter
- bind
- datetimepicker
- div
- head
- submit
- tabbedpanel
- textarea
- tree
- treenode



## VALIDACIONES CON STRUTS

# Validator

- Validator:

- Las validaciones en Struts 2 pueden realizarse a través de ficheros XML, anotaciones o manualmente.
- La validación depende de los interceptores *validation* y *workflow* (ambos incluidos en la pila de interceptores por defecto).
  - El interceptor *validation* realiza la validación y crea la lista de errores.
  - El interceptor *workflow* comprueba si existen errores en la lista, devolviendo *result* input si es así (es necesario definir una vista para el *result*).

<http://struts.apache.org/docs/validation.html>

# Validaciones con Struts

Las validaciones se pueden realizar de forma:

- **Programativa o Manual**

Definiendo e implementando la función que va a realizar la validación.

- **Declarativa**

Definiendo las validaciones que se aplican a cada campo mediante un archivo de configuración o anotaciones

# Validaciones con Struts

- Validación programativa:
  - Incluir un método `public void validate()` en el Action.
  - Utilizar el método `addFieldError` para hacer llegar los mensajes de error a la vista (JSP).

```
public void validate() {  
    if (name.length() < 10) {  
        addFieldError("name", "Longitud del nombre  
        incorrecta");  
    } else if (name.startsWith("A")) {  
        addFieldError("name", "El nombre no puede comenzar por  
        la letra A");  
    }  
}
```

# Validaciones con Struts

- Validación programativa:
  - Agregar el result correspondiente en el fichero de configuración de struts (struts.xml)

```
<result name="input">/example/HelloWorld.jsp</result>
```

# Validaciones con Struts

- Validación declarativa:

- Indicando el result de salida de validacion:

- ```
<result name="input">/index.jsp</result>
```

- Indicar en el JSP de salida dónde se mostrarán los mensajes:

- ```
<s:actionerror />
```

- Indicar en el form si se desea validar en el cliente o en el servidor:

- ```
<s:form ... validate="true">(true= cliente)(recordar añadir <s:head/> en la cabecera)
```

- Crear el fichero de validación.

- NombreAction-validation.xml

- Ubicado en el mismo package que el Action.

# Validaciones con Struts

- Validación declarativa (ejemplo validador):

```
<validators>
  <field name="name">
    <field-validator type="requiredstring">
      <param name="trim">true</param>
      <message>Identificador de usuario
obligatorio</message>
    </field-validator>
  </field>
  <field name="password">
    <field-validator type="requiredstring">
      <param name="trim">false</param>
      <message>Contraseña obligatoria</message>
    </field-validator>
  </field>
</validators>
```

<http://struts.apache.org/development/2.x/docs/validation.html>

# Struts



## Ejemplo 2: Validación



# Struts

- Validación:

- Crear el formulario:

```
<%@taglib prefix="s" uri="/struts-tags" %>
```

```
<s:form action="intentaLogin" method="POST">
```

```
    <s:textfield name="name" label="Login name"/>
```

```
    <s:password name="password" label="Password"/>
```

```
    <s:submit value="Login" align="center"/>
```

```
</s:form>
```

# Struts

- Validación:

- Crear el action:

```
private String name="";
private String password="";
public String execute() throws Exception {
    if (name.equals("admin") && (password.equals("idontknow"))) {
        return SUCCESS;
    } else {
        addActionError("Las credenciales no son correctas");
        return ERROR;
    }
}
```

# Struts

- Validación:

- Registrar el action en el fichero struts.xml:

```
<package name="actions" namespace="/" extends="struts-default">
  <action name="intentaLogin" class="actions.LoginAction">
    <result>/loginok.jsp</result>
    <result name="error">/loginko.jsp</result>
  </action>
</package>
```

- Crear los ficheros loginok.jsp y loginko.jsp

# Struts



## Ejemplo 2.1: Validación con mensaje de error

# Struts

- Validación con mensaje de error:
  - Agregar en el index.jsp del formulario la siguiente línea (mostrará el mensaje de error):  
`<s:actionerror />`
  - Si fuese necesario agregar también:  
`<%@taglib prefix="s" uri="/struts-tags" %>`
  - Modificar el fichero struts.xml:

```
<result>/loginok.jsp</result>  
<result name="error">/index.jsp</result>
```

# Struts



Ejemplo 2.2: Validación de campos de formulario  
(desde el servidor y desde el cliente)

# Struts

- Validación con mensaje de error:

- Indicar el destino en caso de error de validación en **struts.xml**:

```
<action name="intentaLogin" class="actions.LoginAction">  
    <result name="input">/index.jsp</result>  
    <result name="error">/index.jsp</result>  
    <result>/loginok.jsp</result>  
</action>
```

- Indicar en el formulario si se desea hacer la validación en el cliente (**validate="true"**) o en el servidor (**validate="false"**):

```
<s:form action="intentaLogin" method="POST" validate="true">
```

```
<s:form action="intentaLogin" method="POST" validate="false">
```

# Struts

- Validación con mensaje de error :
  - Crear el fichero de validación:
  - Nombre: *NombreAction-validation.xml*
  - Ubicación: junto al Action.

```
<!DOCTYPE validators PUBLIC
"-//OpenSymphony Group//XWork Validator 1.0.2//EN"
"http://struts.apache.org/dtds/xwork-validator-1.0.2.dtd">
<validators>
  <field name="name">
    <field-validator type="requiredstring">
      <param name="trim">true</param>
      <message>Identificador de usuario obligatorio</message>
    </field-validator>
  </field>
  <field name="password">
    <field-validator type="requiredstring">
      <param name="trim">false</param>
      <message>Contraseña obligatoria</message>
    </field-validator>
  </field>
</validators>
```



# Struts

- Validación con mensaje de error:
  - Resultado.

## Formulario de registro

Identificador de usuario obligatorio

Login name:

Contraseña obligatoria

Password:

Login

- Identificador de usuario obligatorio
- Contraseña obligatoria

# Validación declarativa por fichero de configuración

- **Validator:** dos tipos principales de sintaxis
  - *Plain-Validator / Non-Field validators:* permite validar campos y expresiones:

```
<validators>
  <validator type="email">
    <param name="fieldName">myEmail</param>
    <message>Debe proporcionar un email</message>
  </validator>
</validators>
```

```
<validator type="expression">
  <param name="expression">!(password eq '12345')</param>
  <message>Contraseña demasiado obvia</message>
</validator>
```

# Validación declarativa por fichero de configuración

- Validator: dos tipos principales de sintaxis.
  - Field-validator: sólo válida para evaluar campos:

```
<field name="myEmail">  
  <field-validator type="email">  
    <message>Debe proporcionar un email</message>  
  </field-validator>  
</field>
```

# Validación declarativa por fichero de configuración

- Tipos de validators incluidos en el framework (1):
  - conversion validator - Conversión al tipo del atributo del Action.
  - date validator - Valida si la fecha indicada está dentro de un determinado rango.
  - double validator - Valida si el double indicado está dentro de un determinado rango.
  - email validator - Validación de email.
  - expression validator - Validación de una expresión OGNL (Object-Graph Navigation Language). Si no se cumple la expresión, provoca el error.
  - fieldexpression validator - Validación de un campo mediante expresión Ognl.
  - int validator - Valida si el int indicado está dentro de un determinado rango.

<http://struts.apache.org/release/2.3.x/docs/validation.html>

# Validación declarativa por fichero de configuración

- Tipos de validators (y 2):

- regex validator - Validación de un campo mediante una expresión regular.
- required validator - Comprueba si el campo especificado es nulo.
- requiredstring validator - Comprueba si el campo especificado es nulo y mayor de cero (ver API).
- short validator - Valida si el short indicado está dentro de un determinado rango.
- stringlength validator - Valida la longitud de un string.
- url validator - Valida si un String es una URL válida.
- visitor validator - Utilizado para delegación de la validación.
- conditionalvisitor validator - Utilizado para delegación de la validación.

<http://struts.apache.org/release/2.3.x/docs/validation.html>

# Validación declarativa por fichero de configuración (Ejemplo 1/2)

```
<validators>
<field name="bar">
<field-validator type="required">
<message>You must enter a value for bar.</message>
</field-validator>
<field-validator type="int">
<param name="min">6</param>
<param name="max">10</param>
<message>
bar must be between ${min} and ${max}, current value is {bar}.
</message>
</field-validator>
</field>
<field name="bar2">
<field-validator type="regex">
<param name="regex">[0-9],[0-9]</param>
<message>The value of bar2 must be in the format "x, y",
where x and y are between 0 and 9
</message>
</field-validator>
</field>
```

# Validación declarativa por fichero de configuración(Ejemplo 2/2)

```
<field name="date">
  <field-validator type="date">
    <param name="min">12/22/2002</param>
    <param name="max">12/25/2002</param>
    <message>
      The date must be between 12-22-2002 and 12-25-2002.
    </message>
  </field-validator>
</field>
<field name="foo">
  <field-validator type="int">
    <param name="min">0</param>
    <param name="max">100</param>
    <message key="foo.range">Could not find foo.range!</message>
  </field-validator>
</field>
<validator type="expression">
  <param name="expression">foo gt bar </param>
  <message>
    Foo must be greater than Bar. Foo = ${foo}, Bar = ${bar}.
  </message>
</validator>
</validators>
```

# Validación declarativa por fichero de configuración( tipo expression)

```
<field name="password">
  <field-validator type="requiredstring">
    <message key="error.password.required"/>
  </field-validator>
</field>
<field name="password2">
  <field-validator type="requiredstring">
    <message key="error.password2.required"/>
  </field-validator>

  <validator type="expression">
    <param name="expression">password.equals(password2)</param>
    <message key="error.password.match"/>
  </validator>
```

Expresiones OGNL : <http://commons.apache.org/proper/commons-ognl/language-guide.html>



# Validación en cliente

```
<s:form action="Login" validate="true">
  <s:textfield key="username" />
  <s:password key="password" />
  <s:submit/>
</s:form>
```

- Struts2 genera código JavaScript de validación en función de sus etiquetas.

El código Struts2:

```
<form namespace="/example" id="Login"
name="Login"
onsubmit="return validateForm_Login();"
action="/struts2-loginclientsidevalidation/
example/Login.action"
method="post"><table class="wwFormTable">
```

...

Struts2 genera:

```
<script type="text/javascript">
function validateForm_Login() {
form = document.getElementById("Login");
clearErrorMessages(form);
clearErrorLabels(form);
...
```

# Validación declarativa por anotaciones

- Validación:
  - Mediante anotaciones:
    - Indicando las reglas de validación mediante annotations antes de la declaración del método.

```
import com.opensymphony.xwork2.validator.annotations.RegexFieldValidator;

@RegexFieldValidator(
    regex = "[0-9]",
    message = "Por favor, introduce una contraseña válida"
)
public String getPassword() {
    return password;
}
```

<http://struts.apache.org/release/2.3.x/docs/annotations.html>



Struts

# INTERNACIONALIZACIÓN

# Struts

- Internacionalización:
  - A través de ficheros `.properties`
    - Pares clave, valor
    - Múltiples ámbitos:
      - clase Action
      - package
      - struts
      - global
      - ... otros
    - Sintaxis: `nombre_códigoidioma.properties`
      - `global_en.properties`.
      - `struts_it.properties`

<http://struts.apache.org/release/2.3.x/docs/localization.html>

# Struts

- Internacionalización:

- El usuario seleccionará el idioma a través del parámetro `request_locale` que tomará el valor del idioma (ISO-3166).
- Para mostrar las properties (multi-idioma) en lugar del texto existen varias opciones

- `<s:property value="getText('some.key')" />`

- `<s:text name="some.key" />`

- `<s:il8n name="some.package.bundle">`

- `<s:text name="some.key" />`

- `</s:il8n>`

- `<s:textfield key="some.key" name="textfieldName"/>`

<http://struts.apache.org/release/2.3.x/docs/localization.html>

# Struts



## Ejemplo 3: Aplicación multilenguaje mediante selección

# Struts

- Creación del jsp de selección:

```
<s:url id="url" action="SelectLanguageAction">
    <s:param name="request_locale">es</s:param>
</s:url>
<s:a href="%{url}">Español</s:a>
<li>
<s:url id="url" action="SelectLanguageAction">
    <s:param name="request_locale">en</s:param>
</s:url>
    <s:a href="%{url}">English</s:a>
</li>
```

# Struts

- Creación del action (SelectLanguageAction):
  - No hay que programar nada. El Action almacena el atributo locale pasado como parámetro. A partir de la ejecución de este Action el atributo locale tiene el valor indicado.

```
public class SelectLanguageAction extends ActionSupport {  
    public SelectLanguageAction() {  
    }  
    public String execute() throws Exception {  
        return Action.SUCCESS;  
    }  
}
```



# Struts

- Declaración del Action en el fichero struts.xml:

```
<action name="SelectLanguageAction"  
        class="test.SelectLanguageAction">  
    <result>/formulario.jsp</result>  
</action>
```

# Struts

- Creación del formulario (formulario.jsp):

```
<body>
    <h1><s:text name="g.formulario"/></h1>
    <s:form action="VerDatos">
        <s:text name="g.nombre"/>
        <s:textfield name="nombreUsuario"></s:textfield>
        <s:submit/>
    </s:form>
</body>
```

# Struts

- Declaración del uso de fichero de constantes en el fichero `struts.xml`:

```
<struts>
<constant name="struts.custom.i18n.resources" value="global" />
...
</struts>
```

# Struts

- Crear los ficheros de properties (globales) en el package:

- global.properties
- global\_en.properties
- global\_es.properties

con las propiedades:

- g.formulario (Formulario de información)
- g.nombre (Nombre)

# Struts



- Ejecutar.
- Ampliar el ejercicio, implementando el Action “VerDatos” que deberá mostrar los datos introducidos por el usuario en los idiomas de la aplicación.

# Struts

- Internacionalización:

- A las propiedades se puede acceder con las siguientes expresiones:
  - <s:text name="global.saludo"/>
  - <s:property value="getText('global.saludo')"/>
- Los ficheros de properties global.properties irán en el directorio raíz del classpath.
- Los ficheros de propiedades package.properties irán en el directorio del package correspondiente.
- Se puede especificar un fichero de properties por Action.
- Los ficheros de properties pueden estar ubicados en cualquier directorio del classpath y tener cualquier nombre. Se accederá a ellos de la siguiente manera:

```
<spring:message name="paquete.nombrefichero">  
    <br><s:text name="clave_propiedad"/><br>  
</spring:message>
```

# Enlaces

- <http://struts.apache.org/>
- <http://struts.apache.org/release/2.3.x/docs/home.html>
- <http://struts.apache.org/release/2.3.x/docs/tutorials.html>
- <http://struts.apache.org/release/2.3.x/docs/guides.html>
- <http://struts.apache.org/release/2.3.x/docs/validation.html>