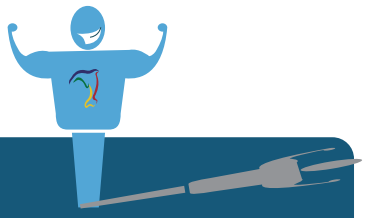
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

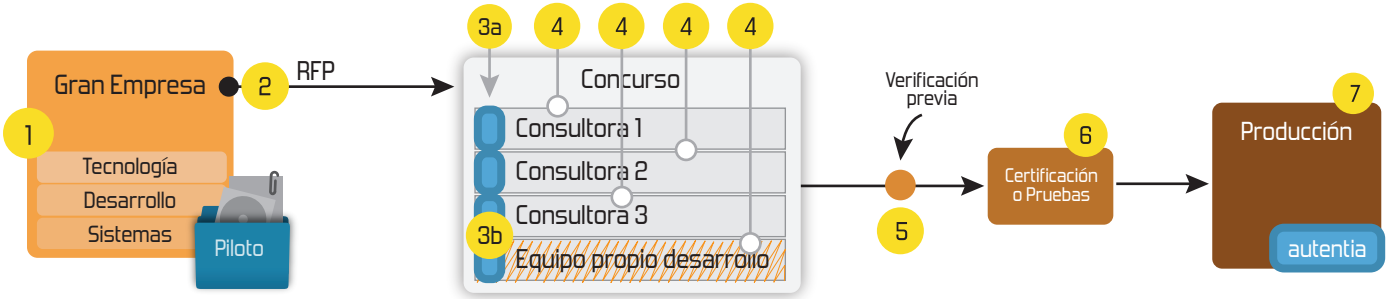
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha aquí...

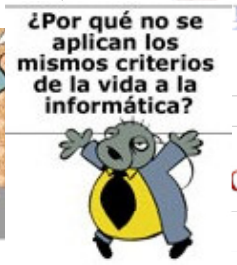
Ver cursos que ofrece Autentia

Descargar comics en PDF y alta resolución

[¡NUEVO!] 2008-12-01

2008-11-17

2008-09-01



2008-07-31

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la seccion comic.

Tutorial desarrollado por

Carlos García Pérez

Creador de MobileTest, un complemento educativo para los profesores y sus alumnos.

Consultor tecnológico en el desarrollo de proyectos informáticos.

Ingeniero Técnico en Informática *

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Catálogo de servicios de Autentia

Descargar (6,2 MB)

Descargar en versión comic (17 MB)

AdictosAlTrabajo.com es el Web de difusión de conocimiento de Autentia.

autentia
real business solutions

Catálogo de cursos

Descargar este documento en formato PDF: struts_appdemo.pdf

Fecha de creación del tutorial: 2009-01-14

Desarrollo de aplicaciones web con Struts 1

En este tutorial y bajo petición de unas personas que me lo solicitaron, voy a intentar plasmar las ideas y los pasos a seguir para que para realizar una pequeña aplicación ejemplo que contemple las ideas y conceptos más importantes en el desarrollo de aplicaciones Web con el FrameWork Struts (1.3.10).

Cabe destacar que la versión 1.3.10, fue la última de las versiones de Struts 1, antes de que naciera el framework Struts 2, cuyo estudio queda pendiente para otro tutorial.

Este tutorial no trata de presentar una extensa documentación teórica sobre este framework, sino más bien, un ejemplo que incluya y explique los conceptos más importantes:

1. Captura de pantalla de la apariencia gráfica de la aplicación a desarrollar.
2. Pasos necesarios para crear un proyecto con Struts (sin usar Maven).
3. Explicación del archivo de configuración de Struts. (`/WEB-INF/struts-config.xml`).
4. JSP de ejemplo: TagLibs de Struts e invocación de acciones (`/index.jsp`).
5. Referencias.

Captura de pantalla de la apariencia gráfica de la aplicación a desarrollar:

Sí, si, la aplicación es muy fea pero resume los conceptos más importantes (configuración, flujo, internacionalización, validación declariva, validación programática, etc.) con dependencias mínimas (2 acciones, 2 formularios, 1 Bean y 3 JSP).

Catálogo de servicios Autentia (PDF 6,2MB)

comic...

oogle™

Web

www.adictosaltrabajo

Buscar

Últimos tutoriales

2009-01-14
Desarrollo de aplicaciones Web con Struts 1

2009-01-07
Log4J: Cómo crear un log que trabaje hacia una Base de Datos.

2009-01-05
Introducción a Google Chart API

2009-01-02
OSCache: Sistema de caché para aplicaciones Java

2008-12-31
Primeros pasos con Selenium IDE

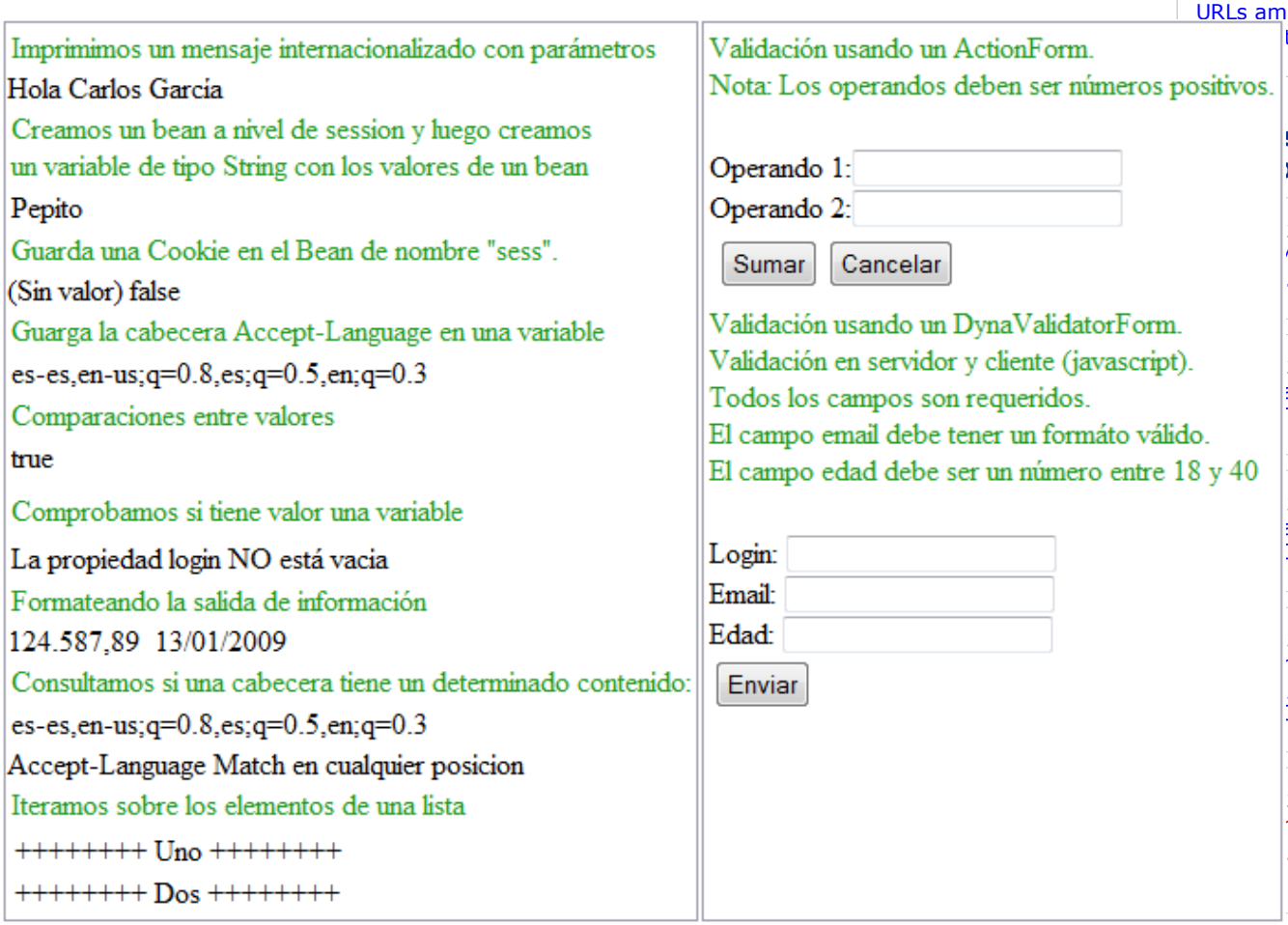
2008-12-29
El componente outputChart de ICEfaces

2008-12-27
JMeter: Tests de rendimiento usando varios clientes distribuidos

2008-12-25
Análisis de rendimiento al usar un Pool de conexiones

2008-12-16
Utilización de grupos en Spring Security

2008-12-17



Puede descargarse la aplicación en formato WAR (con el código fuente) haciendo clic aquí.

Pasos necesarios para crear un proyecto con Struts (sin usar Maven).

1. Descargarnos la distribución de Struts desde la página de descarga oficial.
2. Descomprimir el archivo y copiar todas las librerías (archivos jar) al directorio `/WEB-INF/lib` de tu proyecto web.
3. Dar de alta el servlet controlador de struts en el `/WEB-INF/web.xml` de tu proyecto. Haga clic aquí para verlo.

Archivo `/WEB-INF/web.xml`

```
view plain  print  ?
01. <?xml version="1.0" encoding="UTF-8" ?>
02. <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "ht
    //java.sun.com/dtd/web-app_2_3.dtd">
03. <web-app id="WebApp_ID">
04.     <display-name>StrutsExamples</display-name>
05.     <servlet>
06.         <servlet-name>action</servlet-name>
07.         <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
08.         <init-param>
09.             <param-name>config</param-name>
10.             <param-value>/WEB-INF/struts-config.xml</param-value>
11.         </init-param>
12.         <load-on-startup>1</load-on-startup>
13.     </servlet>
14.
15.     <servlet-mapping>
16.         <servlet-name>action</servlet-name>
17.         <url-pattern>*.do</url-pattern>
18.     </servlet-mapping>
19.
20.     <welcome-file-list>
21.         <welcome-file>index.jsp</welcome-file>
22.     </welcome-file-list>
23. </web-app>
```

Si observa, el servlet controlador de Struts `(org.apache.struts.action.ActionServlet)` tratará todas las peticiones cuya extensión en la URI sea `.do` (do » hacer, aunque esto por supuesto se puede cambiar sin problemas).

Explicación del archivo de configuración de Struts (`/WEB-INF/struts-config.xml`).

Struts se configura de forma declarativa mediante un archivo xml.
En el indicamos básicamente:

- El flujo de la aplicacion, es decir, las posibles acciones a realizar y los caminos a los que pueden conducir cada una de ellas.
- Que hacer en caso de errores.
- El archivo de internacionalización de nuestra aplicación Web.
- Los tipos de datos de la información que es enviada y validada en el método validate los formularios asocionados a cada acción.

Archivo `/WEB-INF/struts-config.xml`

view plain print ?

```
01. <?xml version="1.0" encoding="iso-8859-1"?>
02. <!DOCTYPE struts-config PUBLIC
03.         "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
04.         "http://struts.apache.org/dtds/struts-config_1_3.dtd">
05. <struts-config>
06.     <form-beans>
07.         <form-bean name="operandosForm"  type="com.form.SumarForm"/>
08.         <form-bean name="registroForm" type="org.apache.struts.validator.DynaValidat
09.               <form-property name="login"    type="java.lang.String"></form-
    property>
10.               <form-property name="email"    type="java.lang.String"></form-
    property>
11.               <form-property name="age"      type="java.lang.Byte"></form-
    property>
12.         </form-bean>
13.     </form-beans>
14.
15.     <action-mappings>
16.         <action path="/sumar"
17.                 type="com.action.SumarAction"
18.                 name="operandosForm"
19.                 scope="request"
20.                 validate="true"
21.                 cancellable="true"
22.                 input="/index.jsp">
23.             <forward name="ok"         path="/index.jsp"/>
24.             <forward name="cancel" path="/WEB-INF/jsp/cancel.jsp"/>
25.         </action>
26.
27.         <action path="/registro"
28.             type="com.action.RegistroAction"
29.             name="registroForm"
30.             scope="session"
31.             validate="true"
32.             input="/index.jsp">
33.             <forward name="ok"   path="/WEB-INF/jsp/registerOk.jsp"/>
34.         </action>
35.     </action-mappings>
36.
37.     <message-resources parameter="ApplicationResources" null="false"/>
38.
39.     <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
40.         <set-property
41.                 property="pathnames"
42.                 value="/org/apache/struts/validator/validator-rules.xml, /WEB-INF
    /validation.xml" />
43.     </plug-in>
44. </struts-config>
```

- Línea 06: Dentro del tag `form-beans` se definen los formularios que usa la aplicación. Un formulario es un conjunto de datos que automáticamente Struts nos inicializa con los datos que el usuario ha introduccido en el formulario HTML generado desde la JSP. Existen tres tipos principales de formularios:
  1. `org.apache.struts.action.ActionForm`: Formulario que tiene un atributo, un getter y un setter por cada input del formulario asociada a la acción que lo usa.
     Este tipo de formularios, pueden ser validados de **forma programática**.
     Observe el ActionForm del ejemplo haciendo clic aquí.
  2. `org.apache.struts.action.DynaActionForm`: A diferencia de los `ActionForm` no necesitan que implementemos una clase Java con sus atributos y sus get y set para cada uno de sus atributos. todo se lee y escribe como un mapa (clave atributo/valor atributo), definiéndose previamente en el archivo `struts-config.xml` las propiedades que tendrá (líneas 9-11).
     No permiten validaciones a no ser que nos escribamos una clase que herede de

DynaActionForm y reescribamos el método validate (Por lo que no tiene mucho sentido usarlas si se necesita validación).

3. `org.apache.struts.validator.DynaValidatorForm`: Es un DynaActionForm cuyas validaciones están definidas de **forma declarativa** mediante un archivo xml. Haga clic aqui para ver el archivo de validaciones del ejemplo.

- Línea 07: Definimos "operandosForm" un `SumarForm` (hereda de `ActionForm`) que será usado por la acción "sumar" (línea 18) de manera que antes de ejecutar la acción `SumarAction` se ejecutará el método validate del formulario `SumarForm` y sólo si este método no detecta ningún error, será ejecutada la acción.
- Línea 08: Definimos "registroForm" un `DynaValidatorForm` que será usado por la acción "registro" (línea 27) de manera que antes de ejecutar la acción `RegistroAction` se ejecutarán las validaciones especificadas en el archivo `/WEB-INF/validation.xml` y sólo si no se detecta ningún error, será ejecutada la acción.
- Línea 37: Definimos el nombre del archivo proporcionará la internacionalización de mensajes de por defecto.
  Puede definir archivos para lenguajes y paises específicos, ejemplos:
  - `ApplicationResources_es_ES`: Para el lenguaje Español de España.
  - `ApplicationResources_es`: Para el lenguaje Español en general.
  - `ApplicationResources_es_MX`: Para el lenguaje Español de México.
  - `ApplicationResources_en_US`: Para el lenguaje Inglés de Estados Unidos.
  - `ApplicationResources_en`: Para el lenguaje Inglés en general.
  - etc.

  Haga clic aqui para ver el archivo de internacionalización de mensajes de la aplicación.

- Línea 39: Habilitamos el plugin necesario para realizar validaciones de forma declarativa (sin necesidad de programar, con un archivo xml) gracias al commons Validator. Haga clic aqui para ver el archivo de validaciones del ejemplo.

`ApplicationResources.properties`

```
# ------------------------------------------------------------------------------
# Archivo de internacionalización por defecto: ApplicationResources.properties
# ------------------------------------------------------------------------------

next.page=Siguiente
index=Hola
numeric=No es un número positivo
field.name=Nombre
field.email=Email
field.age=Edad
message.hello=Hola {0}
errors.cancel=cancelada
errors.header=<ul class="errorHeader">
errors.prefix=<li>
errors.suffix=</li>
errors.footer=</ul>
errors.integer={0}: Debe ser un número
errors.required=El campo {0} es requerido
errors.range=El campo {0} debe estár entre {1} y {2}
errors.email=El campo {0} no es un email válido
error.operando.negativo=El operando debe ser un número positivo
error.operandos.negativos=Los operandos deben ser números positivos
```

JSP de ejemplo: TagLibs de Struts e invocación de acciones. (`/index.jsp`).

Dentro de la propia página JSP explico que voy haciendo en cada punto.

view plain print ?

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1" pageEncoding="ISO-8859-1"%>

<%-- Indicamos que vamos a usar las TagLibs de Struts --%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ 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:html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Ejemplo de uso de TagLib de Struts</title>
    <style type="text/css">
        *           { margin:  1px; padding: 1px; }
        .borde      { border-style:solid; border-color:#9999aa; border-
width:1px; }
        div.errores { color: red; font-weight: bold; }
        p.nota      { color: #229922; }
        .errorHeader { background-color:red; color:white;}
    </style>
</head>
<body>
    <%-- Validación usando un ActionForm --%>
    <table>
        <tr style="vertical-align:top">
            <td class="borde">
                <%-- Imprimimos un mensaje internacionalizado con parámetros--%>
                <p class="nota">Imprimimos un mensaje internacionalizado con parámetr
                <bean:message key="message.hello" arg0="Carlos García"  />

                <%-- Creamos un bean a nivel de session y luego creamos un variable c
                <p class="nota">Creamos un bean a nivel de session y luego creamos<br
                <jsp:useBean  id="usuario2"      scope="session" class="com.entity.Us
                <jsp:setProperty name="usuario2" property="login" value="Pepito"/>
                <jsp:setProperty name="usuario2" property="email" value="email@de_pep
                <jsp:setProperty name="usuario2" property="age"   value="24"/>

                <bean:define   id="login"        name="usuario2" property="login"/>
                <p><bean:write name="login" /></p>

                 <%-- Guarda una Cookie en el Bean de nombre "sess". Imprimimos algun
                <p class="nota">Guarda una Cookie en el Bean de nombre "sess".</p>
                <bean:cookie id="sess" name="JSESSIONID" value="(Sin valor)"/>
                <bean:write name="sess" property="value" />
                <bean:write name="sess" property="secure" />

                <%-- Guarga la cabecera Accept-Language en una variable --%>
                <p class="nota">Guarga la cabecera Accept-
Language en una variable</p>
                <bean:header id="var1" name="Accept-Language"/>
                <p><bean:write name="var1" /></p>

                <%-- Comparaciones entre valores --%>
                <p class="nota">Comparaciones entre valores</p>
                <p><logic:equal            name="sess" property="name"   value="JSESS
                <p><logic:notEqual     name="sess" property="name"   value="JSESSION
                <p><logic:greaterThan  name="sess" property="maxAge" value="0">El tie

                <%-- Comprobamos si tiene valor una variable --%>
                <p class="nota">Comprobamos si tiene valor una variable</p>
                <p><logic:empty        name="usuario2" property="login">La propiedad
                <p><logic:notEmpty name="usuario2" property="login">La propiedad logi

                <%-- Formateando la salida de información --%>
                <p class="nota">Formateando la salida de información</p>
                <%
                    pageContext.setAttribute("unDouble", new Double(124587.89));
                    pageContext.setAttribute("ahora",    new java.util.Date(System.cu
                %>
                <bean:write name="unDouble" format=",000.00" />  <bean:write name="ah
/MM/yyyy" />

```

/index.jsp (Continuación)

```
view plain  print  ?
01.     <%-- Consultamos si una cabecera tiene un determinado contenido (también vale par
02.     <p class="nota">Consultamos si una cabecera tiene un determinado contenido:</p>
03.     <bean:header id="language" name="Accept-Language"/>
04.     <p><bean:write name="language"/></p>
05.     <logic:present header="Accept-Language" >
06.         <logic:match     header="Accept-Language" value="es">Accept-
Language Match en cualquier posicion</logic:match>
07.         <logic:notMatch header="Accept-Language" value="es">Accept-
Language notMatch en cualquier posicion</logic:notMatch>
08.     </logic:present>
09.     <logic:notPresent header="Accept-Language">Accept-
Language no existe</logic:notPresent>
10.
11.     <%-- Iteramos sobre los elementos de una lista --%>
12.     <p class="nota">Iteramos sobre los elementos de una lista</p>
13.    <%
14.         java.util.ArrayList list = new java.util.ArrayList();
15.         list.add("++++++++ Uno ++++++++");
16.         list.add("++++++++ Dos ++++++++");
17.         pageContext.setAttribute("lista", list);
18. %>
19.         <ol>
20.              <logic:iterate id="element" name="lista" scope="page">
21.                  <li><bean:write name="element" /></li>
22.              </logic:iterate>
23.          </ol>
24. </td>
```

/index.jsp (Continuación)

```
view plain  print  ?
            <td class="borde">
                <%-- Nota: Los operandos deben ser números positivos. (Validación usa
                <p class="nota">
                    Validación usando un ActionForm.
                    <br/>Nota: Los operandos deben ser números positivos.
                </p>
                <html:form action="sumar"  method="post" focus="operando1" >
                    <br/>Operando 1:<html:text property="operando1"/> <html:errors pr
                    <br/>Operando 2:<html:text property="operando2"/> <html:errors pr
                    <table>
                        <tr>
                            <td><html:submit value="Sumar"/></td>
                            <td><html:cancel value="Cancelar"/></td>
                        </tr>
                    </table>

                    <logic:notEmpty name="operandosForm" property="resultado">
                        <p> <strong>El resultado es</strong>:
                            <bean:write name="operandosForm" property="resultado"/>
                        </p>
                    </logic:notEmpty>
                </html:form>

                <%-- Validación usando un DynaValidatorForm con validación tanto en e
                <p class="nota">Validación usando un DynaValidatorForm.
                    <br/>Validación en servidor y cliente (javascript).
                    <br/>Todos los campos son requeridos.
                    <br/>El campo email debe tener un formáto válido.
                    <br/>El campo edad debe ser un número entre 18 y 40
                </p>

                <%-- Indicamos que genere el código JavaScript necesario para validar
             (Las validaciones las optione del archivo validation.xml) --%>
                <html:javascript formName="registroForm" />
                <html:form action="registro"  method="post" focus="login" onsubmit="r
                    <br/>Login: <html:text property="login"/> <html:errors property='
                    <br/>Email: <html:text property="email"/> <html:errors property='
                    <br/>Edad:  <html:text property="age"/>   <html:errors property='
                    <p><html:submit value="Enviar"/></p>
                </html:form>
            </td>
        </tr>
    </table>

    <%-- En caso de que existan errores, los mostramos --%>
    <logic:messagesPresent>
        <div class="errores">
            <html:errors/>
        </div>
    </logic:messagesPresent>
</body>
</html:html>
```

/WEB-INF/jsp/cancel.jsp

Si observa el archivo struts-config.xml verá que está JSP es mostrada cuando el usuario cancela la acción de sumar.

```
view plain  print  ?
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1" pageEncoding="ISO-8859-1"%>
<!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>Cancelada</title>
</head>
<body>
    Cancelada
</body>
</html>
```

/WEB-INF/jsp/registerOk.jsp

Si observa el archivo struts-config.xml verá que está JSP es mostrada cuando el usuario se registra

correctamente a través de la acción de registro.

```
view plain  print  ?
01. <%@ page language="java" contentType="text/html; charset=ISO-
    8859-1" pageEncoding="ISO-8859-1"%>
02. <%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
03. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org
    /TR/html4/loose.dtd">
04. <html>
05. <head>
06.     <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
07.     <title>Página de confirmación</title>
08. </head>
09. <body>
10.     <h1>Â¡Â¡ Se ha registrado correctamente !!</h1>
11.     <h2>Datos de registro</h2>
12.     <ul>
13.         <li>Login: <bean:write name="registerUser" property="login"/></li>
14.         <li>Email: <bean:write name="registerUser" property="email"/></li>
15.         <li>Edad:  <bean:write name="registerUser" property="age"/></li>
16.     </ul>
17. </body>
18. </html>
```

com.entity.User

```
view plain  print  ?
01. package com.entity;
02.
03. /**
04.  * Bean para el ejemplo de struts => representa un usuario
05.  * @author Carlos García. Autentia.
06.  * @see http://www.mobiletest.es
07.  */
08. public class User {
09.     private String login;   // Código del usuario
10.     private String email;   // Dirección de correo
11.     private byte   age;     // Edad
12.
13.     public String getLogin() {
14.         return login;
15.     }
16.     public void setLogin(String login) {
17.         this.login = login;
18.     }
19.     public String getEmail() {
20.         return email;
21.     }
22.     public void setEmail(String email) {
23.         this.email = email;
24.     }
25.     public byte getAge() {
26.         return age;
27.     }
28.     public void setAge(byte age) {
29.         this.age = age;
30.     }
31. }
```

com.form.SumarForm

Observe que tiene un atributo y un get y set para cada uno de ellos. Además estos deben de coincidir en nombre con los input del formulario HTML

```
view plain  print  ?
01. package com.form;
02.
03. import org.apache.commons.validator.GenericValidator;
04. import org.apache.struts.action.ActionErrors;
05. import org.apache.struts.action.ActionMessage;
06.
07. /**
08.  * ActionForm usado para la acción "sumar"
09.  * @author Carlos García. Autentia.
10.  * @see http://www.mobiletest.es
11.  */
12. public class SumarForm extends org.apache.struts.action.ActionForm {
13.     private String operando1;
14.     private String operando2;
15.     private String resultado;
16.
17.     public String getOperando1() {
18.         return operando1;
19.     }
20.
21.     public void setOperando1(String operando1) {
22.         this.operando1 = operando1;
23.     }
24.
25.     public String getOperando2() {
26.         return operando2;
27.     }
28.
29.     public void setOperando2(String operando2) {
30.         this.operando2 = operando2;
31.     }
32.
33.     public String getResultado() {
34.         return resultado;
35.     }
36.
37.     public void setResultado(String resultado) {
38.         this.resultado = resultado;
39.     }
40.
41.     /*
42.      * Inicializamos los parámetros
43.      */
44.     public void reset(org.apache.struts.action.ActionMapping mapping, javax.servlet.H
45.         this.operando1 = "";
46.         this.operando2 = "";
47.         this.resultado = "";
48.     }
49.
50.     /*
51.      * Validamamos los datos introducidos por el usuario
52.      */
53.     public ActionErrors validate(org.apache.struts.action.ActionMapping mapping, java
54.         ActionErrors    errors      =  new ActionErrors();
55.
56.         if (GenericValidator.isBlankOrNull(this.operando1) || (! GenericValidator.isI
57.             errors.add("operando1", new ActionMessage("error.operando.negativo"));
58.         }
59.
60.         if (GenericValidator.isBlankOrNull(this.operando2) || (! GenericValidator.isI
61.             errors.add("operando2", new ActionMessage("error.operando.negativo"));
62.         }
63.
64.         return errors;
65.     }
66. }
67.
68.
```

com.action.SumarAction

En caso de que el método validate de SumarForm no detecte errores se ejecutará esta acción y se redirigirá a la página `/index.jsp`.
Todo esto está definido en el `action` con el atributo `path="sumar"` dentro del archivo struts-config.xml

```
view plain  print  ?
01. package com.action;
02.
03. import javax.servlet.http.HttpServletRequest;
04. import javax.servlet.http.HttpServletResponse;
05. import org.apache.struts.action.*;
06.
07. /**
08.  * Action que suma dos numeros enteros positivos
09.  * @author Carlos García. Autentia.
10.  * @see http://www.mobiletest.es
11.  */
12. public class SumarAction extends Action {
13.
14.     public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServlet
15.         ActionForward forward = null;
16.
17.         if (this.isCancelled(request)){
18.             forward = mapping.findForward("cancel");
19.         } else {
20.             com.form.SumarForm operationForm = (com.form.SumarForm) form;
21.
22.             int operando1 = Integer.parseInt(operationForm.getOperando1());
23.             int operando2 = Integer.parseInt(operationForm.getOperando2());
24.
25.             operationForm.setResultado(String.valueOf(operando1 + operando2));
26.
27.             forward = mapping.findForward("ok");
28.         }
29.
30.         return forward;
31.     }
32. }
```

com.action.RegistroAction

Registra al usuario a partir de los datos introducidos en la JSP `/index.jsp` y redirige a la página `/WEB-INF/jsp/registerOk.jsp` donde serán mostrados los datos del usuario.
Todo esto está definido en el `action` con el atributo `path="registro"` dentro del archivo struts-config.xml

```
view plain  print  ?
01. package com.action;
02.
03. import javax.servlet.http.HttpServletRequest;
04. import javax.servlet.http.HttpServletResponse;
05. import org.apache.struts.action.*;
06.
07. /**
08.  * Action que registra al usuario en el sistema
09.  * @author Carlos García. Autentia.
10.  * @see http://www.mobiletest.es
11.  */
12. public class RegistroAction extends Action {
13.
14.     public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServlet
15.         DynaActionForm registerForm = (DynaActionForm) form;
16.
17.         // Extraemos los datos del formulario
18.         String  login = (String) registerForm.get("login");
19.         String  email = (String) registerForm.get("email");
20.         Byte    edad  = (Byte)   registerForm.get("age");
21.
22.         ////////////////////////////////////////////////////////////////////
23.         // Realizariamos las acciones en los objetos de negocio (En este caso,
24.         ////////////////////////////////////////////////////////////////////
25.
26.         // Configurariamos los objetos Request, Session, etc. que necesita la
27.         com.entity.User user = new com.entity.User();
28.         user.setLogin(login);
29.         user.setEmail(email);
30.         user.setAge(edad.byteValue());
31.
32.         request.setAttribute("registerUser", user);
33.
34.         // Mostramos la siguiente vista
35.         return mapping.findForward("ok");
36.     }
37. }
```

/WEB-INF/validation.xml

En este archivo especificamos las validaciones de forma declarativa de los formularios `form-beans` definidos en el `/WEB-INF/struts-config.xml`

Las validaciones a realizar son:

- Los campos login, email y edad son requeridos.
- El campo email debe tener un formato de email válido.
- El campo edad debe ser un número comprendido entre 18 y 40.

view plain print ?

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE form-validation PUBLIC
     "-//Apache Software Foundation//DTD Commons Validator Rules Configuration 1.3.0/
     "http://jakarta.apache.org/commons/dtds/validator_1_3_0.dtd">
<form-validation>
    <formset>
        <form name="registroForm">
            <field property="login" depends="required">
                <arg key="field.name"/>
            </field>

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

            <field property="age" depends="required, integer, intRange">
                <arg position="0" key="field.age"/>
                <arg position="1" name="intRange" key="${var:min}" resource="false"/:
                <arg position="2" name="intRange" key="${var:max}" resource="false"/:
                <var>
                    <var-name>min</var-name>
                    <var-value>18</var-value>
                </var>
                <var>
                    <var-name>max</var-name>
                    <var-value>40</var-value>
                </var>
            </field>
        </form>
    </formset>
</form-validation>
```

Referencias:

- Guía de referencia de Struts (en Inglés)
- Guía de validación con Struts (en Inglés)

Un saludo.
Carlos García. Creador de MobileTest, un complemento educativo para los profesores y sus alumnos.

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de subcripción a novedades:

E-mail Aceptar

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	pdf
Integración de Spring con el envío de emails: técnicas avanzadas (II)	Tercer y último tutorial de la serie sobre envío de email a través del soporte que proporciona Spring. Veremos cómo generar el contenido del mensaje, de forma dinámica, mediante plantillas de Velocity	2008-12-09	505	-	pdf
Como crear y destruir programáticamente un RMI Registry	En este sencillo tutorial vamos a ver como podemos arrancar o parar el Registry programáticamente, es decir lo arrancaremos y pararemos desde el propio programa Java.	2008-02-07	1394	-	pdf
Activar el soporte SSL en Struts	Os mostramos las particularidades de uso y configuración de Struts para trabajar con SSL	2005-12-06	10910	-	pdf
JSF y comparativa con Struts	Os mostramos los pasos necesarios para empezar a utilizar JSF (Java Server Faces) y su comparación / relación con Struts	2005-09-08	33343	-	pdf
SpringIDE, plugin de Spring para Eclipse	En adictosaltrabajo os hemos ido presentando diversos plugins para Eclipse. Esta vez le toca el turno a SpringIDE, un plugin que os ayudará a desarrollar aplicaciones que utilicen Spring.	2008-01-19	5553	-	pdf
Extender la validación en Struts	Os mostramos con un ejemplo como extender los mecanismos de validación en Struts, utilizando el framework Commons Validator	2006-02-17	13954	-	pdf
Análisis de rendimiento al usar un Pool de conexiones	Análisis de rendimiento de usar o no un pool de conexiones a bases de datos en nuestras aplicaciones	2008-12-25	519	-	pdf
URLs amigables con UrlRewriteFilter	Análisis de un filtro a nivel de aplicación web que permite la sobreescritura de URLs para la construcción de URLs amigables	2008-12-17	502	-	pdf
Múltiples struts-config.xml e internacionalización de Jasper Report	En este tutoral queremos mostraros como dividir el struts-config.xml en dos o más ficheros, pero sin usar la capacidad de módulos que tiene struts y como generar informes con la ayuda de Jasper Report con un contenido diferente dependiendo del idioma del	2007-03-19	13784	-	pdf
Introducción a Struts Flow	Struts Flow es un módulo de extensión del conocido framework Struts, que facilita la implementación del flujo de páginas de una aplicación web	2006-01-02	8770	-	pdf

Nota: