

¿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

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

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

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

Tutorial desarrollado por: [Beatríz Bonilla](#)

Puedes encontrarme en [Autentia](#)
Somos expertos en Java/J2EE
Contacta en info@autentia.com



Descargar este documento en formato PDF [jstlJSF.pdf](#)

[Firma en nuestro libro de Visitas](#)

[Master Java J2ee Oracle](#)

Prácticas laborales 100% aseguradas Nuevo temario de Struts. Trabaja ya

[IntelliJ IDEA](#)

Advanced JSP Editor for professional developers. Get Trial

[JSP Editor](#)

Edit JSP, XML, DTD, Schema, XSLT & SOAP. Easy-to-Use! Free Trial.

[Formación Empresas](#)

Consultoría de Formación Tecnologías Web

UTILIZANDO JSTL EN APLICACIONES JSF

Introducción

Java Server Pages Standard Tag library (JSTL) es un conjunto de librerías de etiquetas simples y estándares que encapsulan funcionalidad habitualmente utilizada en aplicaciones Web. Soportan funciones como iteraciones, procesamiento condicional, procesamiento XML, internacionalización, etc. Su uso está extendido debido sobre todo a que se integran en las páginas JSP de forma sencilla y limpia, además de proporcionar la mayoría de funcionalidades necesarias en un JSP.

Por otra parte *Java Server Faces* (JSF) es un marco de trabajo basado en el patrón MVC (*Modelo-Vista-Controlador*) cuya característica novedosa con respecto a Struts, por ejemplo, es que la parte de *Vista* se crea a partir de un conjunto de componentes reutilizables disponibles del lado del servidor.

En este tutorial veremos algunos beneficios y también problemas al intentar utilizar estas dos tecnologías combinadamente.

Herramientas utilizadas

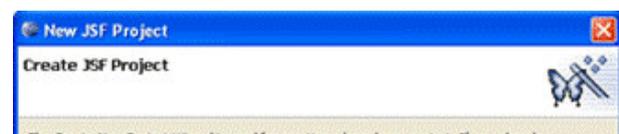
- Sistema operativo: Windows XP Professional.
- Servidor Web: Apache TomCat 5.5.9
- Entorno de desarrollo: Eclipse 3.1.1 con ExadelStudio-3[1].0.5

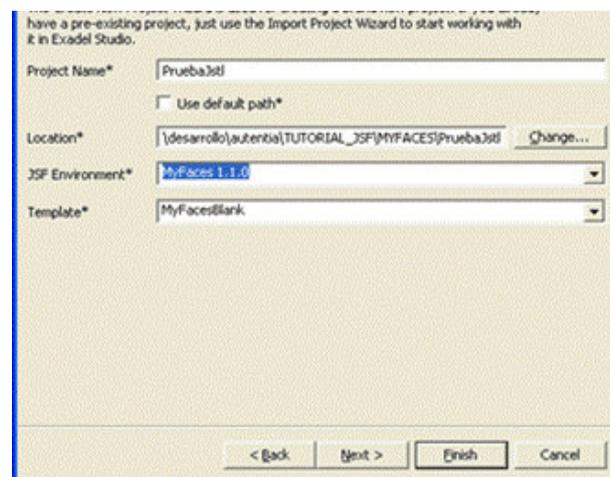
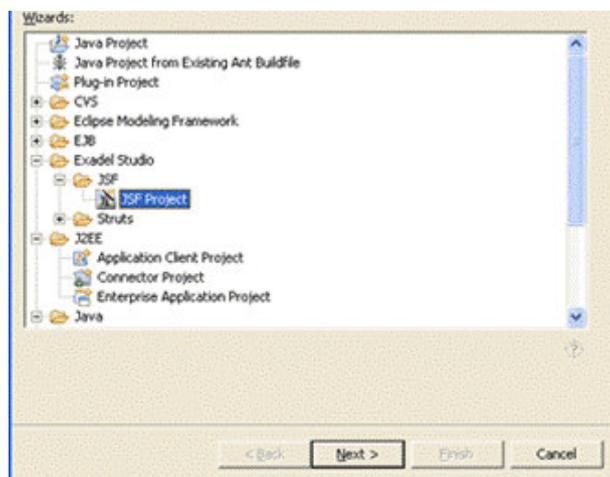
Preparamos el entorno

Vamos a seguir los siguientes pasos:

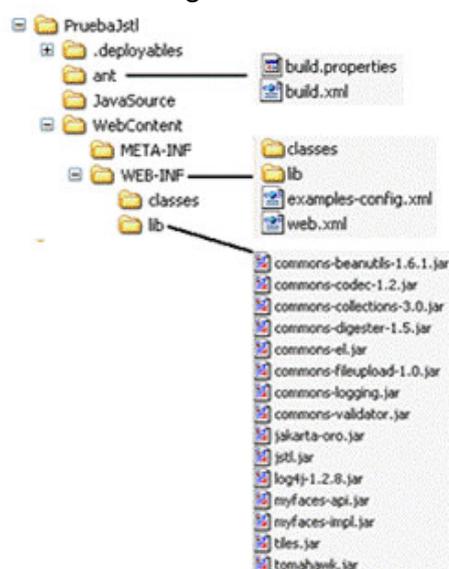
1. Creamos un nuevo proyecto JSF en Eclipse.

Para ello selecciono *File>New>Project>JSF Project*





Los ficheros generados son los siguientes:



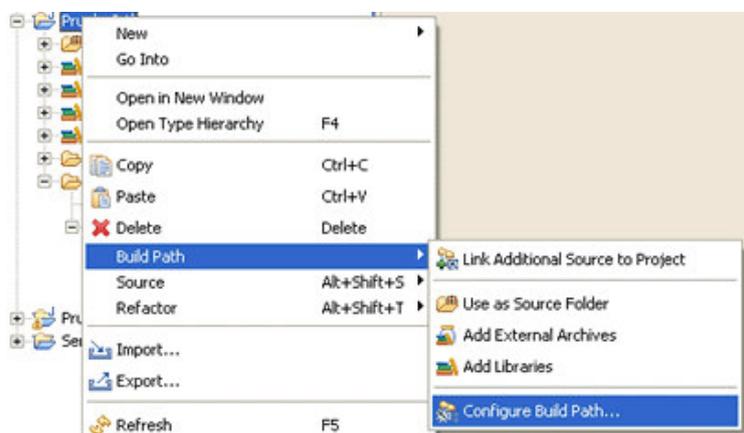
2. Incorporamos al proyecto todo lo necesario para trabajar con JSTL

Debemos conseguir las librerías y taglibs de JSTL; todo esto lo podemos conseguir en:

<http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html>

A día de hoy, la versión más actual se encuentra en el fichero *jakarta-taglibs-standard-20060227.zip*; lo descomprimos y copiamos los ficheros *jstl.jar* y *standard.jar* al directorio *lib* de nuestra aplicación. La librería *jstl.jar* ya la había incorporado Exadel por defecto al crear el proyecto, pero por evitar problemas de versiones, he machacado la que ya había.

Después, actualizamos el classpath del proyecto con las nuevas librerías



Además de las librerías, es necesario incorporar a nuestro proyecto los taglib que serán invocados en nuestras páginas JSF. El único que voy a utilizar para este ejemplo es *c.tld*, así

que lo copio al directorio *WEB-INF* de la aplicación.

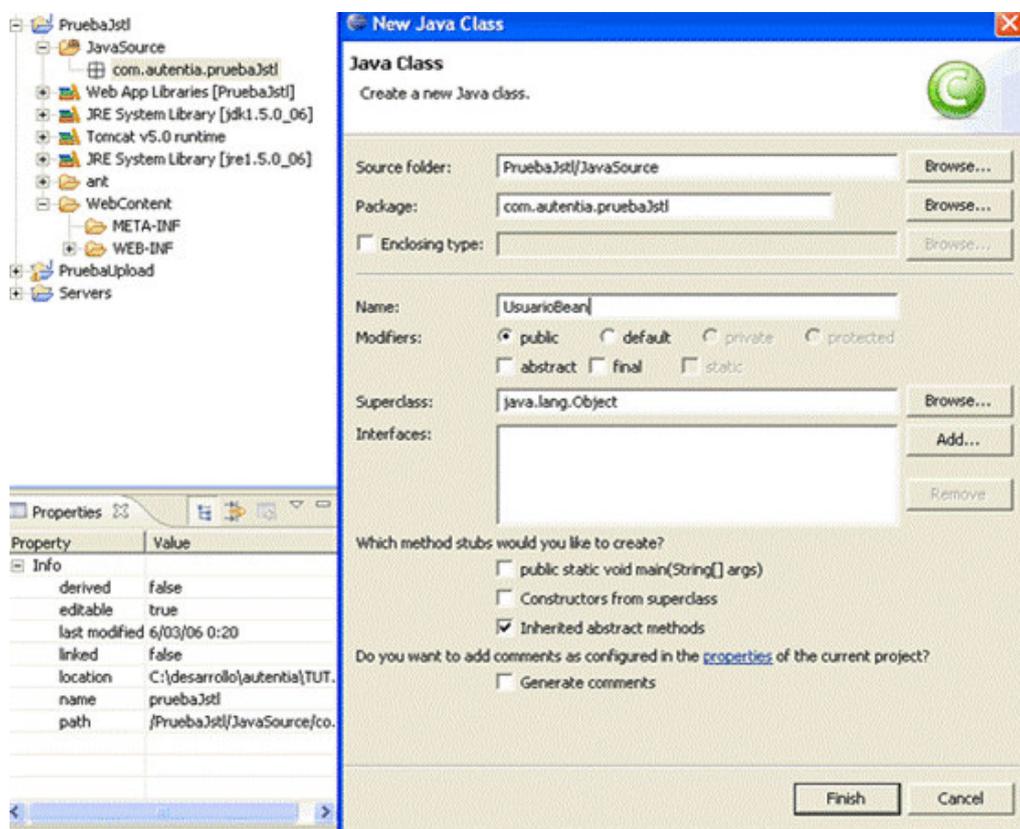
3. Creamos la página *index.jsp*

Esta página redireccionará a la página *gestionUsuarios.jsf* que crearemos más adelante:

```
<%@ taglib uri="http://myfaces.apache.org/extensions" prefix="t" %>
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>AUTENTIA- TUTORIAL JSTL-JSF</head>
  <body>
    <jsp:forward page="gestionUsuarios.jsf" />
  </body>
</html>
```

4. Generando el bean y las clases de apoyo

Creamos el bean *UsuarioBean* cuya responsabilidad, por ahora, será la de proporcionar un listado estático de usuarios. Para ello, sobre el paquete *JavaSource* seleccionamos *New>Class* y creamos la clase:



Creamos las variables de instancia que necesitamos, y preparamos un método que devuelve un listado de usuarios:

```
package com.autentia.pruebaJstl;
import java.util.ArrayList;
import java.util.List;
/**
 * Bean encargado de la gestión de usuarios
 * @author AUTENTIA
 */
public class UsuarioBean implements Serializable
{
    /** Login del usuario */
    private String login;
    /** Password del usuario */
    private String password;
    /** Nombre completo del usuario */
    private String nombre;
    /** Lista de usuarios */
```

```

private static List listaUsuarios;
////////// CONSTRUCTORES
public UsuarioBean()
{
    listaUsuarios = new ArrayList();
}
////////// FIN DE CONSTRUCTORES
////////// METODOS DE ACCESO A VARIABLES PRIVADAS
public String getLogin() {
    return login;
}
public void setLogin(String login) {
    this.login = login;
}
public String getNombre() {
    return nombre;
}
public void setNombre(String nombre) {
    this.nombre = nombre;
}
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
}
public List getListaUsuarios()
{
    if(listaUsuarios.size()==0)
        listaUsuarios=creaListaInicialUsuarios();
    return listaUsuarios;
}
//////////FIN DE METODOS DE ACCESO A VARIABLES PRIVADAS
/**
 * Crea una lista estática de usuarios.
 */
private static List creaListaInicialUsuarios()
{
    List IResultados = new ArrayList();
    IResultados.add(new Usuario("autentia1","pautentia1","Javier Pérez
Gómez"));
    IResultados.add(new Usuario("autentia2","pautentia2","Juan Cuesta
García"));
    IResultados.add(new Usuario("autentia3","pautentia3","Pepe González
Cam"));
    return IResultados;
}
}

```

Nos apoyamos en una clase que representa la entidad usuario:

```

package com.autentia.pruebaJstl;
/**
 * Clase que representa un usuario
 * @author AUTENTIA
 */
public class Usuario implements Serializable
{
    /** Login del usuario */
    private String login;
    /** Password del usuario */
    private String password;
    /** Nombre del usuario */
    private String nombre;
}

```

```

    /** Constructor */
    public Usuario(String login,String password,String nombre)
    {
        this.login=login;
        this.nombre=nombre;
        this.password=password;
    }
    public String getLogin() {
        return login;
    }
    public void setLogin(String login) {
        this.login = login;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}

```

5. Registramos el bean en el fichero descriptor de JSF

en nuestro caso, el fichero descriptor se llama `examples-config.xml`:

```

<faces-config>
  <managed-bean>
    <description>
      Bean para la gestión de usuarios
    </description>
    <managed-bean-name>usuariosBean</managed-bean-name>
    <managed-bean-class>com.autentia.pruebaJstl.UsuarioBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
</faces-config>

```

6. Creamos los ficheros de recursos

Así los mensajes que aparezcan en nuestra aplicación se adaptan al idioma del usuario que esté accediendo a la misma; en este tutorial sólo los voy a generar en español e inglés; estos ficheros estarán en el paquete `com.autentia.pruebaJstl`:

recursos_es	recursos_en
<pre> ##### FICHEROS DE RECURSOS usuarios_bienvenido=Bienvenido!!! gestionUsuarios_generacion_jsf=Listado generado con JSF gestionUsuarios_generacion_jstl=Listado generado con JSTL gestionUsuarios_nombre=Nombre gestionUsuarios_login=Login </pre>	<pre> ##### BUNDLES FILE usuarios_bienvenido=Welcome!!! </pre>

Primera prueba. Mostrando la lista de usuarios con el componente dataTable de JSF y con JSTL

Creamos la página `gestionUsuarios.jsp` y mostramos la lista de usuarios con JSF como se muestra a continuación:

```

<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://myfaces.apache.org/extensions" prefix="t" %>
<html>
<head>
<title>AUTENTIA - TUTORIAL JSTL-JSF</title>
<style>
.tabla {font-family : verdana;
        font-size: 12px;
        padding: 0;
        cellpadding:0;
        cellspacing:0;
        color: #000000;width: 80%;}
.cabecera_tabla {font-family: Verdana;
                 font-size: 11px;
                 color: #000000;
                 background-color: #AFBDB6;
                 font-weight:bold;padding: 0;cellpadding:0;cellspacing:0;
                 text-align: left;border: 1 px;}
.fila {background-color: #FFFFE0;}
.columna_left {text-align: left;width:50%}
.texto {font-family : verdana;
        font-weight:bold; font-size: 12px;color: #000000;}
</style>
</head>
<body>
<f:loadBundle basename="com.autentia.pruebaJstl.recursos" var="mensajes"/>
<h:form id="gestionUsuarios">
<f:view>
<center>
<h:outputText
    value="#{mensajes['gestionUsuarios_generacion_jsf']}"
    styleClass="texto" />
<f:verbatim><br><br></f:verbatim>
<t:dataTable id="data" styleClass="tabla"
             headerClass="cabecera_tabla"
             footerClass="cabecera_tabla" rowClasses="fila"
columnClasses="columna_left,columna_left" var="usuario"
             value="#{usuarioBean.listaUsuarios}"
preserveDataModel="true" rows="10">
    <h:column>
    <f:facet name="header">
        <h:outputText value="#{mensajes['gestionUsuarios_nombre']}" />
    </f:facet>
    <t:commandLink action="#{usuarioBean.preparaParaEditar}"
        immediate="true">
        <h:outputText value="#{usuario.nombre}" />
    </t:commandLink>
    </h:column>
    <h:column>
    <f:facet name="header">
    <h:outputText value="#{mensajes['gestionUsuarios_login']}" />
    </f:facet>
        <h:outputText value="#{usuario.login}" />
    </h:column>
    </t:dataTable>
</center>
</f:view>
</h:form>
</body>
</html>

```

Con:

```
<f:loadBundle basename="com.autentia.pruebaJstl.recursos" var="mensajes"/>
```

cargamos el fichero de recursos y con `t:dataTable` visualizamos la lista de usuarios que proporciona el método `getListaUsuarios` del bean `usuariosBean`.

Probamos en el Tomcat lo que llevamos hecho y perfecto...

Listado generado con JSF

Nombre	Login
Javier Pérez Gómez	autentia1
Juan Cuesta García	autentia2
Pepe González Cam	autentia3

Ahora intentamos hacer lo mismo pero utilizando JSTL; para ello importamos el taglib correspondiente:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://myfaces.apache.org/extensions" prefix="t" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
.....
```

Y creamos el listado:

```
....
<br><br>
<center>
<h:outputText
  value="#{mensajes['gestionUsuarios_generacion_jstl']}" styleClass="texto" />
<table class="tabla">
  <tr>
  <td class="cabecera_tabla">
    <h:outputText value="#{mensajes['gestionUsuarios_nombre']}" />
  </td>
  <td class="cabecera_tabla">
    <h:outputText value="#{mensajes['gestionUsuarios_login']}" />
  </td>
  </tr>
  <c:forEach items="${sessionScope.usuarioBean.listaUsuarios}" var="usuario1">
  <tr class="fila">
  <td class="columna_left">
    <t:commandLink action="#{usuarioBean.preparaParaEditar}" immediate="false">
      <c:out value="${usuario1.nombre}"/>
    </t:commandLink>
  </td>
  <td class="columna_left"><c:out value="${usuario1.login}"/></td>
  </tr>
  </c:forEach>
  </table>
</center>
</f:view>
</h:form>
</body>
</html>
```

Como vemos, accedo al bean a través de la sesión:

```
<c:forEach items="${sessionScope.usuarioBean.listaUsuarios}" var="usuario1">
```

Cuando despliego en Tomcat me encuentro el primer problema: parece que los `commandLink` no funcionan con JSTL:

Listado generado con JSF

Nombre	Login
Javier Pérez Gómez	autentia1
Juan Cuesta García	autentia2
Pepe González Cam	autentia3

Listado generado con JSTL

Nombre	Login
Javier Pérez Gómez	autentia1

Juan Cuesta García
Pepe González Cam

usuario1
autentia2
autentia3

Esto es debido a que el `<c:forEach>` es evaluado por el contenedor de JSP cada vez que se procesa la página; el contenedor deja disponible en la página (*Page scope*) la variable `usuario1`, pero el concepto de *Page scope* es específico de JSP, y no existe en los entornos que maneja JSF. Este problema está documentado en la especificación de JSF (sección 9.2.8).

Parece entonces que los taglibs de JSTL sí pueden acceder a los beans JSF que dejamos en sesión, en el request o a nivel de aplicación, pero no al contrario.

Segunda prueba. Creando/Editando un usuario

La página de creación y edición de los datos de un usuario será la misma, aunque en función del tipo de acción, habrá que hacer las siguientes consideraciones:

- Si estamos creando un nuevo usuario, no tiene sentido que aparezca el botón de eliminar al usuario
- Si estamos editando los datos del usuario, el campo login no puede modificarse.

Estas excepciones las voy a manejar con JSTL en este ejemplo, aunque con JSF sería mucho más adecuado utilizar el atributo *rendered* que tienen todos los componentes precisamente para esto, que se muestren o no.

Antes, en la página `gestionUsuarios.jsp` añadido un enlace para crear un nuevo usuario:

```
....
....
<f:view>
  <table class="tabla">
    <tr>
      <td align="right">
        <t:commandLink action="#{usuarioBean.nuevoUsuario}" immediate="true">
          <h:outputText value="#{mensajes['nuevo_usuario']}" />
        </t:commandLink>
      </td>
    </tr>
  </table>
....
....
```

Y completo el bean `usuarioBean` para que cuando pinchemos en *Nuevo usuario* o sobre uno de los usuarios existentes, nos lleve a la página de edición:

```
...
...
/** Indicador de si se puede mostrar el botón de borrar */
private String mostrarBotonBorrar;
/** Indicador de si el campo de login es editable */
private String loginEditable;
...
...
/**
 * Muestra el formulario de creación de un usuario
 */
public String nuevoUsuario()
{
    // Limpiamos los campos del formulario
    clear();
    return "edicion_usuario";
}

/**
 * Limpia los campos del bean
 */
private void clear()
{
    this.login="";
    this.nombre="";
    this.password="";
    this.mostrarBotonBorrar="false";
    this.loginEditable="true";
}
```

```

    }

    /**
     * Recupera como parámetro el login del usuario que se desea editar, rellena los
    atributos
     * del bean y redirige a la página de edición de usuarios
     * @return la página de edición de usuarios
     */
    public String preparaParaEditar()
    {
        String donde = null;
        FacesContext context = FacesContext.getCurrentInstance();
        Map map = context.getExternalContext().getRequestParameterMap();
        // Recuperamos el login del usuario que deseamos editar
        String idUsuario = (String) map.get("idUsuario");
        Usuario usuario = obtenerUsuario(idUsuario);
        if(usuario!=null)
        {
            fill(usuario);
            this.mostrarBotonBorrar="true";
            this.loginEditable="false";
            donde = "edicion_usuario";
        }
        return donde;
    }
}

```

```

/**
 * Itera sobre la lista de usuarios y devuelve el que tenga el login pasado como
 * parámetro.
 */
private Usuario obtenerUsuario(String login)
{
    Usuario us = null;
    List lista = getListaUsuarios();
    for(int i=0;i<lista.size();i++)
    {
        Usuario obj = (Usuario)lista.get(i);
        if(obj.getLogin().equalsIgnoreCase(login))
        {
            us = obj;
            break;
        }
    }
    return us;
}

/**
 * Rellena el bean a partir de los datos de un usuario
 * @param usuario
 */
private void fill(Usuario usuario)
{
    this.login=usuario.getLogin();
    this.nombre=usuario.getNombre();
}
}

```

Ahora añadido la navegabilidad en el fichero `examples-config.xml`:

```

<navigation-rule>
  <from-view-id>/gestionUsuarios.jsp</from-view-id>
  <navigation-case>
    <from-outcome>edicion_usuario</from-outcome>
    <to-view-id>/edicionUsuario.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

```

```

    <redirect/>
  </navigation-case>
</navigation-rule>

```

Creamos la página de edición de un usuario, que llamaremos *edicionUsuario.jsp*...

```

<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://myfaces.apache.org/extensions" prefix="t" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <title>AUTENTIA - TUTORIAL JSTL-JSF</title>
    <link rel="stylesheet" type="text/css" href="css/estilos.css">
  </head>
  <body>
    <f:loadBundle basename="com.autentia.pruebaJstl.recursos" var="mensajes"/>
  <f:view>
    <h:form id="usuarioBean" name="usuarioBean">
      <h:panelGrid columns="2" styleClass="gestionUsuariosFormTable"
        headerClass="gestionUsuariosFormHeader"
        footerClass="gestionUsuariosFormFooter"
        columnClasses="gestionUsuariosFormLabels, gestionUsuariosFormInputs"
        width="600">
        <!-- Login -->
        <h:outputLabel for="login" value="Login"/>
        <c:if test="${sessionScope.usuarioBean.loginEditable=='true'}">
          <h:inputText id="login" styleClass="CajasTexto" size="30" maxLength="50"
            value="#{usuarioBean.login}" required="true">
            <f:validateLength maximum="100" minimum="1"/>
          </h:inputText>
        </c:if>
        <c:if test="${sessionScope.usuarioBean.loginEditable=='false'}">
          <h:outputText value="#{usuarioBean.login}"/>
        </c:if>

        <!-- Nombre -->
        <h:outputLabel for="nombre" value="Nombre"/>
        <h:inputText id="nombre" styleClass="CajasTexto" size="30" maxLength="50"
          value="#{usuarioBean.nombre}" required="true">
          <f:validateLength maximum="100" minimum="1"/>
        </h:inputText>
        <h:panelGroup>
          <h:commandButton action="#{usuarioBean.save}"
            value="#{mensajes['bot_salvar']}" />
          <f:verbatim>&nbsp;  </f:verbatim>
          <c:if test="${sessionScope.usuarioBean.mostrarBotonBorrar=='true'}">
            <h:commandButton action="#{usuarioBean.delete}" immediate="true"
              value="Borrar" />
          </c:if>
        </h:panelGroup>
      </h:panelGrid>
    </h:form>
  </f:view>
</body>
</html>

```

Este ejemplo parece que sí funciona. Si seleccionamos el botón *Nuevo Usuario* el sistema permitirá al usuario introducir un nuevo login y no aparece el botón *Borrar*:

The image shows a simple web form with two input fields. The first field is labeled "Login" and the second is labeled "Nombre". Both fields are empty and have a light blue border. The form is centered on a white background.

Y si seleccionamos un usuario del listado, aparecerán los datos editados, no se podrá modificar el login y sí aparecerá el botón de *borrar*, qué bien:

Login autentia2

Nombre

Recuerda

que el personal de [Autentia](#) te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#))

¿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?

info@autentia.com

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

Autentia = Soporte a Desarrollo & Formación

J2EE, EJBs, Struts...

[Autentia S.L.](#) Somos expertos en:
J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
 y muchas otras cosas

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	<input style="width: 90%;" type="text"/>
	<input type="button" value="Enviar"/>

Otros Tutoriales Recomendados ([También ver todos](#))

Nombre Corto	Descripción
Upload de ficheros en JSF	Os mostramos de una forma sencilla y guiada como crear una utilidad de upload de ficheros utilizando JSF
JSF y comparativa con Struts	Os mostramos los pasos necesarios para empezar a utilizar JSF (Java Server Faces) y su comparación / relación con Struts
Upload de ficheros en Struts	En este tutorial os mostramos paso a paso como construir una sencilla aplicación de upload de ficheros utilizando Struts
Evitar doble-click en JSPs y Struts	Os mostramos como construir unas librerías de TAGs para evitar el problema de doble-click en aplicaciones JSP y como se soluciona (qué teneis que hacer) con el framework Struts
Probando entornos para JSF	En este tutorial os mostramos con ejemplos como utlilizar dos conocidos entornos de desarrollo para JSF: Exadel Studio y Sun Studio Creator
Struts y EL en Netbeans 4	Os mostramos como configurar la última contribución de Struts que se integra con el lenguaje de expresiones EL.
Plantear una aplicación Web y Struts	Os mostramos un posible modo de plantear una aplicación Web (análisis) y darla forma. El Framework utilizado es struts y tratamos de identificar qué depende de este Framework y qué no.
Aplicación profesional con Struts	En este tutorial, os mostramos como crear una aplicación profesional, usando numerosos patrones y utilizando Struts.

[Extender la validación en Struts](#)

Os mostramos con un ejemplo como extender los mecanismos de validación en Struts, utilizando el framework Commons Validator

[Desarrollo Struts con XDoclet](#)

Alejandro Perez nos enseña como simplificar el desarrollo de aplicaciones J2EE basadas en Struts, automatizando la generación de código con XDoclet

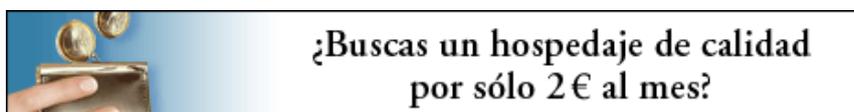
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores.

En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo.

Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.

[Patrocinados por enredados.com Hosting en Castellano con soporte Java/J2EE](#)



www.AdictosAlTrabajo.com Optimizado 800X600