Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

Framework Corporatiu J2EE

Servei de Tags de Presentació

Versió 1.1

Barcelona, 3 / octubre / 2006

Històric de modificacions

Data	Autor	Comentaris	Versió
09/12/2005	Atos Origin, sae openTrends	Versió inicial del document	1.0
29/09/2006	Atos Origin	Versió 1.1	1.1

Llegenda de Marcadors

Índex

1. Introducció

1.1. Propòsit

El Servei de Tags té com a propòsit principal oferir un ventall de components a utilitzar dins de les nostres pàgines. En l'actualitat es basen en l'ús de pàgines JSP (Java Server Pages), per ser ara mateix la tecnologia de presentació més estesa.

openFrame, en el seu afany de proporcionar la màxima flexibilitat i facilitat als desenvolupadors ha seleccionat, estès i millorat algunes propostes de la comunitat. En molts dels components es fan servir tecnologies Web d'última generació i s'estenen solucions openSource ja existents. Tanmateix s'ha definit una política de configuració que facilita l'herència de valors de les propietats entre components, aspecte no assolible amb la tecnologia de tags JSP.

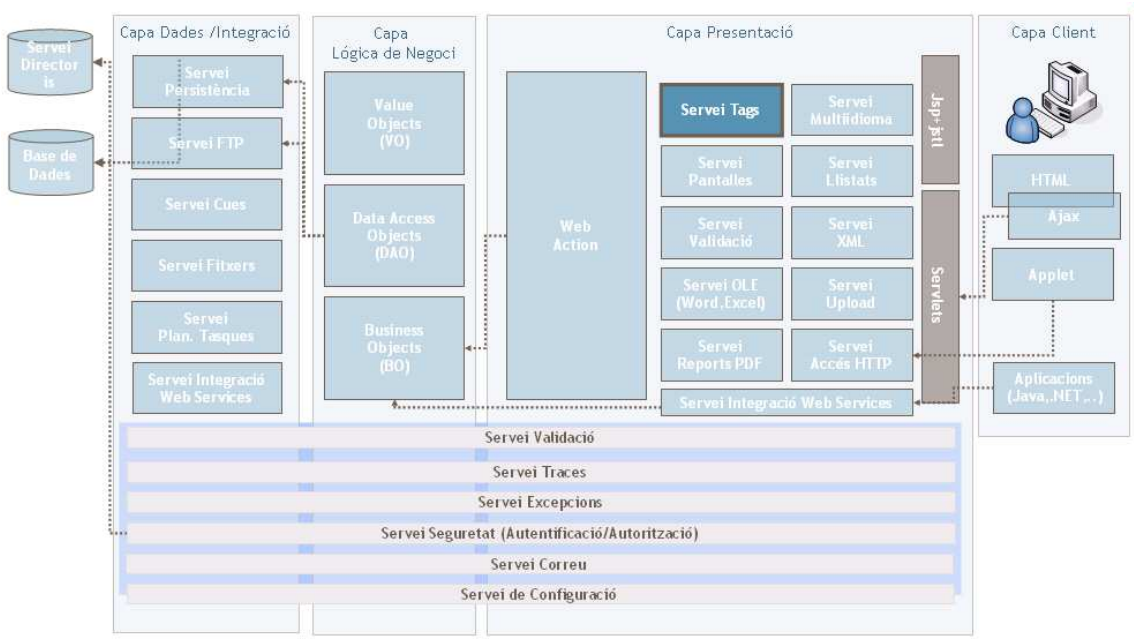
En l'actualitat existeix una nova generació de desenvolupament Web anomenada AJAX [1](**Asynchronous JavaScript And XML)**. Tot i que el desenvolupador no necessita del coneixement d'aquest tecnologia, openFrame s'ha adaptat a aquesta nova generació en la implementació interna dels seus tags.

Es considera prerequisit per a la lectura d'aquest document el coneixement en HTML, CSS i l'ús de Struts.

1.2. Context i Escenaris d'Ús

El Servei de Tags es troba dins dels serveis de Presentació de openFrame.

1 Podeu veure una introducció a 'http://www.adaptivepath.com/publications/essays/archives/000385.php'



Per a la utilització dels tags de openFrame es requereix d'un entorn basat en les següents tecnologies:

- JSP
- JSTL
- Struts

1.3. Versions i Dependències

En el present apartat es mostren quines són les versions i dependències necessàries per fer ús del Servei.

1.3.1 Versions

Novetats en versió 1.1.

- Configuració dels tags en el JSP (Aparatat 2.3)
- Tags Avançats (Apartat 2.4)
- Llistats editables (Apartat 2.5)
- Validacions en el jsp. (Apartat 2.6)

1.3.2 Dependències bàsiques

Nom	Tipus	Versió	Descripció
ajaxtags	jar	1.1	

Nom	Tipus	Versió	Descripció
dwr	jar	1.1.3	
jsp-api	jar	2.0	http://java.sun.com/products/jsp
jstl	jar	1.1.1	
nekohtml	jar	0.8.1	
openFrame-core	jar	1.0	
openFrame-services-configuration	jar	1.0	
openFrame-services-exceptions	jar	1.0	
openFrame-services-i18n	jar	1.0	
openFrame-services-logging	jar	1.0	
openFrame-services-validation	jar	1.0	
openFrame-services-web	jar	1.0	
oro	jar	2.0.8	http://jakarta.apache.org/oro/
js	jar	1.5R4.1	
servlet-api	jar	2.4	
spring	jar	1.2.5	http://www.springframework.org
struts-layout	jar	1.2_20051027	
struts-menu	jar	2.3	
struts	jar	1.2.7	

Per cada servei openFrame fer ús també de les dependències requerides per ell.

Es recomana consultar la referència del servidor d'aplicacions en el que es farà el deployment de l'aplicació per conèixer si les versions aquí indicades estan permeses.

1.4. A qui va dirigit

Aquest document va dirigit als següents perfils:

- Programador. Per configurar els tags i editar les pàgines
- Dissenyador. Per definir els estils i la composició de les pàgines

1.5. Documents i Fonts de referència

Ubicació dels labels	http://illuminosity.net/thoughts/archives/styling_forms/
Llibreria per accessKey	http://www.gerv.net/software/aul/
Llibreria Struts Layout	http://struts.application-servers.com/
Formularis accessibles	http://www.websemantics.co.uk/tutorials/accessible_forms/#texttop
Advanced CSS	http://www.yourhtmlsource.com/stylesheets/advancedcss.html
Overlib	http://www.bosrup.com/web/overlib/
Dom tooltip	http://www.mojavelinux.com/projects/domtooltip/HOWTO.html
Dynarch Calendar	http://www.dynarch.com/projects/calendar/
DHTML Internet Explorer	http://msdn.microsoft.com/workshop/author/dhtml/reference/objects.asp
Prototype library	http://prototype.conio.net/
Value List	http://valuelist.sourceforge.net/
Struts Layout	http://struts.application-servers.com/doc/tags/layout.html
Ditchnet JSP Tags	http://209.61.157.8:8080/taglibs/

1.6. Glossari

TLD (Tag Library Descriptor)

Un TLD és un fitxer en format XML que permet a un desenvolupador definir un conjunt de tags. Per cada tag es poden descriure els seus atributs i la classe associada que generarà el contingut de sortida a la pàgina en la que es faci servir.

Tot i que l'ús de tags específics permet eliminar l'ús de codi Java des de les nostres pàgines JSP, no eliminen la necessitat de crear codi Java. S'han de crear igualment classes Java que facin el treball del tag. Aquestes classes han de estendre les classes definides al package 'tagext' de J2EE.

2. Descripció Detallada

2.1. Arquitectura i Components

L'arquitectura del Servei de Tags és basa en l'ús de llibreries de tags JSP.

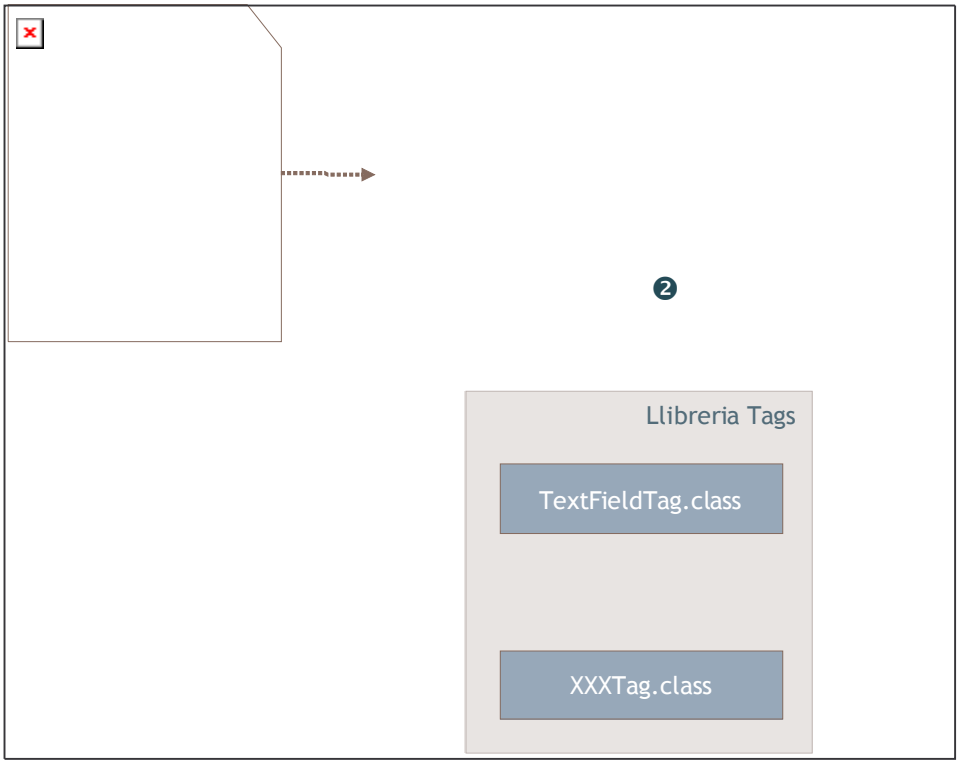
Es recomana una lectura prèvia de la url 'http://java.sun.com/products/jsp/tutorial/TagLibrariesTOC.html' per conèixer què és una llibreria de tags, com es creen i com s'utilitzen.

A mode de resum, l'arquitectura que hi ha al darrera de les llibreries de tags és la mostrada a continuació:

1) El contenidor del Servidor d'Aplicacions en generar el contingut de sortida a partir de la pàgina JSP troba la referència '<%taglib>' on s'indica la localització de la llibreria de tags que resol tot tag de la pàgina que contingui el prefix 'fwk'.

2) Quan troba el tag <fwk:text> accedeix al descriptor tld i troba quina és la classe que generarà el contingut a substituir en la localització del tag

3) El tag realitza la sortida a partir dels atributs passats a la pàgina JSP. Aquest contingut s'incorpora a la pàgina

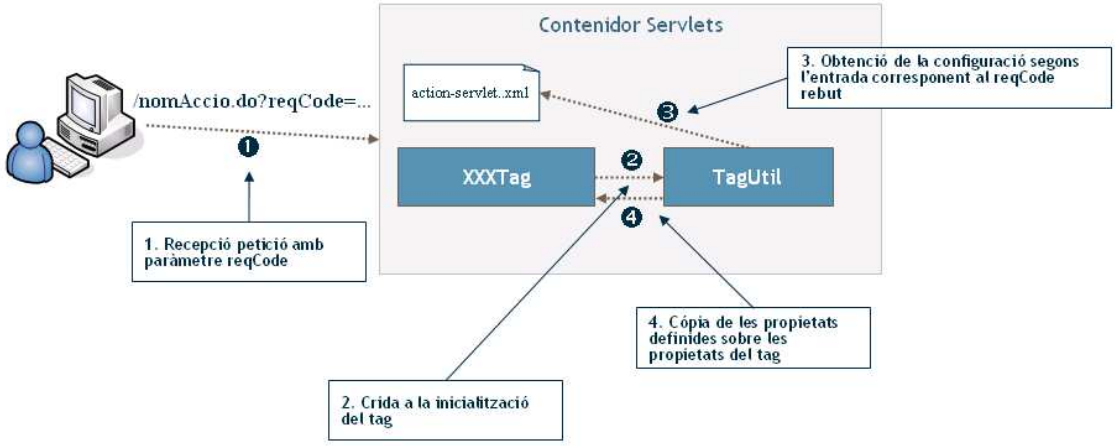


Seguint la filosofia d'independència màxima de la implementació openFrame defineix interfícies per tots els tags. En l'actualitat s'ofereixen implementacions basades en JSP i Struts. Degut a la filosofia que permet configurar de forma externa els tags de les pàgines, la migració a altra tecnologia serà molt més fàcil.

2.1.1. Arquitectura de Procediment d'Inicialització Intern

La configuració dels tags es realitza mitjançant el Servei de Configuració com si de classes planes es tractessin. El model inherent a JSP, pel qual és el contenidor qui instancia els tags no ens permet de fer ús directe del patró 'Dependency Injection'. En qualsevol cas, es fa ús dels mecanismes definits pel Servei de Configuració per a inicialitzar els tags.

El procediment seguit per a la inicialització d'un tag és el següent:



1) Es rep la petició de l'usuari (segons per exemple un submit) en el que arriba un paràmetre 'reqCode' que indica el mètode de l'acció Struts que s'executarà. Una vegada finalitzada l'acció aquesta retorna la pàgina JSP que mostrarà el contingut de la resposta.

2) En la generació del contingut de sortida el contenidor del servidor crida a la inicialització de cadascun dels tags inclosos a la pàgina JSP.

   Imaginem que a la pàgina JSP tenim un component definit de la següent forma:

```
<fwk:text styleId="firstname" property="firstname"/>
```

   És obligatori que definim aquestes 2 propietats a les nostres pàgines. L'atribut 'property' és equivalent al que fa servir Struts, que ens permet lligar quin atribut de l'objecte volem presentar. L'atribut 'styleId' servirà per obtenir la configuració del tag des del fitxer de configuració.

   El propi tag, dins el seu mètode d'inicialització crida a la classe 'TagUtil' indicant-li quin és el seu 'styleId' per obtenir les seves propietats.

3) La classe TagUtil consulta el fitxer de configuració 'action-servlet.xml' i obté el bloc definit dins l'entrada corresponent al 'reqCode' que ha arribat com a paràmetre i les propietats definides per el 'styleId' del tag.

```
<property name="tagsConfiguration">
        <map>
                <entry key="*" ❶>
                        <list>
                        ...
                                <bean id="..." class="...">
```

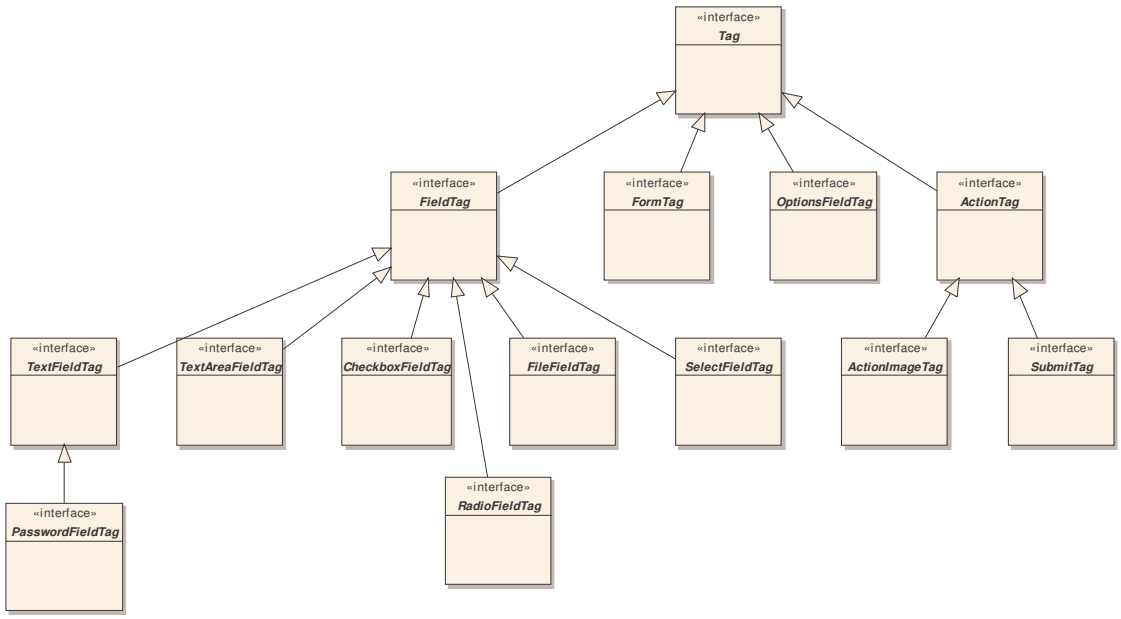
```
                        <property name="styleId"
                    value="nombreEstiloId" ❷/>

                    </bean>
            </list>
    </entry>


    <entry key="reqCode">
            <list>
                    <bean ...>
                        <property name="styleId"
                    value="nombreEstiloId"/>

                    </bean>
            </list>
    </entry>
    </map>
```

❶ Segons el reqCode que arribi com a paràmetre a la url '...?reqCode=...' s'escollirà una entry o una altra per configurar els tags de la pàgina.
❷ L'estil indicat lliga amb l'atribut 'styleId' que s'ha definit a la pàgina JSP

4) El tag, mitjançant commons beanutils copia les propietats definides sobre les seves propietats. A continuació realitza la generació de la sortida segons les propietats definides.

2.1.2. Interfícies i Components Genèrics

Els tags del servei es troben definits mitjançant interfícies que defineixen amb mètodes què s'espera de les implementacions.



En el gràfic mostrat a dalt es pot veure la jerarquia d'interfícies definida per openFrame.

Component	Package	Descripció
Tag	net.opentrends.openframe.services.web.taglib	Interfície base de tots els tags. Defineix principalment la necessitat que totes les implementacions hagin definit mètodes d'accés als atributs: • 'styleId'. Identificador del tag dins la pantalla • 'pageContext'. Obtenció del context de la pàgina • validationService. Referència al servei de validació • i18nService. Referència al servei d'internacionalització
FieldTag	net.opentrends.openframe.services.web.taglib	Interfície de definició de tots els tags que permetin l'entrada de dades en un formulari
FormTag	net.opentrends.openframe.services.web.taglib	Interfície de definició d'un tag que representa un formulari
OptionsFieldTag	net.opentrends.openframe.services.web.taglib	Interfície de definició d'un tag que representi els valors d'una llista d'opcions
ActionTag	net.opentrends.openframe.services.web.taglib	Interfície tags d'acció d'un formulari
TextFieldTag	net.opentrends.openframe.services.web.taglib	Interfície tag entrada d'una dada d'una sola línia en un formulari
TextAreaFieldTag	net.opentrends.openframe.services.web.taglib	Interfície tag entrada de dada de vàries línies en un formulari
CheckboxFieldTag	net.opentrends.openframe.services.web.taglib	Interfície tag de marcar un valor en un formulari
FileFieldTag	net.opentrends.openframe.services.web.taglib	Interfície tag per enviar un fitxer dins un formulari
SelectFieldTag	net.opentrends.openframe.services.web.taglib	Interfície tag de selecció d'una llista de valors
ActionImageTag	net.opentrends.openframe.services.web.taglib	Interfície tag per usar una imatge de botó per acceptar un formulari i enviar-lo
SubmitTag	net.opentrends.openframe.services.web.taglib	Interfície tag per usar una botó per acceptar un formulari i enviar-lo
PasswordFieldTag	net.opentrends.openframe.services.web.taglib	Interfície tag entrada de dades que no es visualitzen per pantalla
RadioFieldTag	net.opentrends.openframe.services.web.taglib	Interfície tag per selecció d'un radio button

2.1.3. Implementació basada en Struts

La implementació basada en Struts de les interfícies mostrades es troben dins el package 'net.opentrends.openframe.services.web.struts.taglib.forms.fields'. Els seus noms coincideixen amb els de les interfícies.

Els tags actuals de openFrame es basen en extensions dels tags de Struts-Layout. Per evitar una màxima dependència amb aquesta implementació s'han definit interfícies de tots els tags (tal i com s'ha vist en l'anterior apartat). Veure l'apartat 'Configuració i Instal·lació' per a més detalls.

2.2 Instal·lació i Configuració

Prerequisit: Per al correcte funcionament d'aquest servei és important que prèviament s'hagin realitzat les configuracions especificades a l'apartat 'Configuració Bàsica' del document 'Serveis de Presentació'.

2.2.1 Instal·lació

Per a la instal·lació del Servei de Tags es requereix de l'ús del fitxer 'openFrame-services-web.jar' i les dependències indicades a l'apartat 'Versions i Dependències'.

2.2.2 Configuració General

A diferència del tradicional mecanisme de tags JSP en el que dins la pàgina s'especifiquen les propietats del tag, openFrame va més enllà i permet tractar els tags com classes Java normals, a fi que es puguin injectar les seves propietats mitjançant el Servei de Configuració.

Aquesta aplicació es fa només pels components d'entrada de formularis, degut a la necessitat que tenen diferenciada segons l'acció realitzada.

openFrame fa ús del Servei de Configuració per a definir els diferents comportaments d'una mateixa plana de presentació segons el cas d'ús. Això implica que una mateixa plana JSP pot tenir diferents configuracions de visualització i comportament dinàmic, descrites de manera declarativa. Les nostres pàgines JSP seran molt simples, ja que s'externalitzaran les seves propietats en aquests fitxers de configuració, en els quals podem utilitzar la potència de l'herència per definir propietats comuns.

Per cada acció Struts configurada al fitxer 'action-servlet.xml' definirem la propietat 'tagsConfiguration' on definirem el següent format:

La configuració del Servei de Tags implica els següents passos:

1) Definir la propietat de configuració de l'acció
2) Definir les propietats de cada tag

Definició de la configuració per l'acció

```
<bean name="/action"
class="....XXXAction">
       ...
       <property name="tagsConfiguration">
              <map>
                     <entry key="*">
                            <list>
```

La definició de les propietats que s'usaran pels tags d'una pantalla es poden definir per cada acció de l'aplicació. Així, per exemple podríem representar en una creació alguns aspectes de forma diferent.
Per definir quines són les propietats dels tags per un reqCode en concret es definirà la propietat 'tagsConfiguration'. Aquesta propietat és un mapa, on:

- La clau es farà servir com a cerca de la configuració segons el 'reqCode' rebut al paràmetre.

  Si es defineix amb el valor '*' es considerà com la configuració per defecte, en tant que s'aplicarà en els següents casos:

  a) No arriba 'reqCode' a la url
  b) Si no es troba configuració d'un determinat tag en la configuració segons el 'reqCode' rebut, es cercarà en la configuració per defecte '*'
  c) Si la definició específica hereta de la definició realitzada a la configuració per defecte '*'

- El valor de l'entrada correspondrà a una llista de beans que definirà les propietats de cadascun dels tags

Exemple:

```
<bean name="/accounts❶"
class="net.opentrends.openframe.samples.jpetstore.struts.action.AccountAction">
   ...
   <property name="tagsConfiguration❷">
          <map>
                 <entry key="*"❸>
                        <list>

                               <bean id="idForm" ...>
                                      <property name="styleId"
                                      value="actionForm"/>
                               </bean>
```

```
                                <bean id="idFirstName" ...>
                                        <property name="styleId"
                                        value="firstname"/>
                                        <property name="accesskey"
      value="forms.accountForm.field.firstname.accesskey"/>

                                        <property name="key"
      value="forms.accountForm.field.firstname"/>                        ...

                                </bean>


                        <entry key="edit"❹>
                                <list>
                                <bean id="idFirstName" parent="textFieldTag">
                                        <property name="styleId"
                                <value="name"/>

                                </bean>
                                </list>
                        </entry>
                        </map>
                </property>
      </bean>
```

❶ Acció de Struts amb injecció de les seves propietats (veure document 'Serveis de Presentació' per a més referència).
❷ Definició de la propietat 'tagsConfiguration'
❸ Per qualsevol reqCode es farà servir la configuració dels tags aquí indicada
❹ En cas que arribés el paràmetre 'reqCode' amb valor 'edit' es farà servir aquesta configuració

Una vegada definida la propietat de configuració de l'acció podem definir les propietats de cadascun dels tags.

Definició de les propietats d'un tag

```
<bean name="/action"
class="....XXXAction">
      ...
      <property name="tagsConfiguration">
            <map>
                  <entry key="*">
                        <list>
                              <bean id="" class="">
                                    ...
                              </bean>
```

La configuració d'un tag és equivalent a la configuració de qualsevol altre element (consultar el document 'Servei de Configuració' per més referència).

A l'atribut 'class' doncs s'especificarà quin és la classe concreta d'implementació del tag que farem servir.

En el cas de Struts farem ús de les següents classes:

- net.opentrends.openframe.services.web.struts.taglib.forms.fields.FormTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.OptionsFieldTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.ActionTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.TextFieldTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.TextAreaFieldTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.CheckboxFieldTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.FileFieldTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.SelectFieldTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.ActionImageTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.SubmitTag
- net.opentrends.openframe.services.web.struts.taglib.forms.fields.PasswordFieldTag

En tots ells és necessari configurar les següents propietats:

Propietat	Requerit	Descripció
styleId	Sí	Identificador del tag. És l'identificador de connexió amb el camp 'styleId' del tag a la pàgina JSP.

Addicionalment, depenent del tag podrem configurar altres propietats. Tots ells permeten configurar les propietats dels tags en el que es basen. Podem per tant configurar qualsevol dels atributs de Struts mitjançant el fitxer de configuració.

La potència real d'aquesta forma de treballar versus els clàssics tags JSP és que podem usar l'herència i definir propietats genèriques a un grup de tags. Així, podem definir propietats comunes a varis tags mitjançant el Servei de Configuració tal i com es mostra en el següent exemple:

```
<!-- TAG's default configuration -->
   <bean id="formTag"
   class="net.opentrends.openframe.services.web.struts.taglib.forms.FormTag"/>

   <bean id="actionImageTag"
class="net.opentrends.openframe.services.web.struts.taglib.forms.ActionImageTag"/>
   <bean id="submitTag"
class="net.opentrends.openframe.services.web.struts.taglib.forms.SubmitTag"/>

   <bean id="textFieldTag"
   class="net.opentrends.openframe.services.web.struts.taglib.forms.fields.TextFie
ldTag">
          <property name="i18nService" ref="i18nService"/>
          <property name="validationService" ref="webValidationService"/>
          <property name="layout" value="true"/>
   </bean>
   ...
```

```
   <bean id="checkboxFieldTag"
class="net.opentrends.openframe.services.web.struts.taglib.forms.fields.CheckboxFi
eldTag">
          <property name="i18nService" ref="i18nService"/>
          <property name="validationService" ref="webValidationService"/>
          <property name="layout" value="true"/>
   </bean>
```

En aquest exemple, es defineixen les propietats comunes als diferents tipus de components. Així podem fer ús de l'herència per definir els components específics de la nostra aplicació com per exemple:

```
<bean parent="textFieldTag">
   <property name="styleId" value="id"/>
   <property name="mode" value="E,I,I"/>
   <property name="key" value="forms.accountForm.field.id"/>

</bean>
```

Aquí s'ha eliminat la necessitat de definir el id del tag i s'ha fet ús de l'herència mitjançant el tag 'textFieldTag' que fa referència al '<bean id=”textFieldTag”>' que s'ha definit en el fitxer extern.

Podem, per últim, estructurar la següent forma el nostre fitxer de l'aplicació:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
   <!-- TAG's default configuration -->

   <bean id="formTag"
   class="net.opentrends.openframe.services.web.taglib.config.FormTagConfiguration
"/>
   <bean id="fieldTag">
         ...
   </bean>

   <bean id="textFieldTag"
   class="net.opentrends.openframe.services.web.taglib.config.TextFieldTagConfigur
ation" parent="fieldTag">
   </bean>


   <import resource="action-servlet-accounts.xml"/>
   <import resource="action-servlet-categories.xml"/>
   <import resource="action-servlet-products.xml"/>
   <import resource="action-servlet-items.xml"/>

</beans>
```

On mitjançant el tag 'import' es fa referència a diferents fitxers de configuració (grups per acció, per una sola acció, etc.)

A continuació s'ofereix explicació de la configuració de cadascun dels tags.

2.2.3 Configuració del Tag 'ConfigurationTag'

Ús: Obligatori, doncs genera les llibreries dels components

El tag Configuration de openFrame s'encarrega de generar totes les llibreries necessàries que requereixen els tags.

Per a generar les llibreries necessàries accedeix al fitxer de propietats de Spring i per cada tag definit obté les seves propietats. Segons els valors d'aquestes propietats genera les llibreries. Per exemple, en el cas de especificar 'showCalendar' a un TextField, generarà les llibreries de calendari, així com la llibreria de literals de calendari de l'idioma de l'usuari entre d'altres.

Atributs:

Atributs	Requerit	Descripció
id	Sí	Usar el valor 'configurationTag'
class	Sí	Usar 'net.opentrends.openframe.services.web.struts.taglib.configuration.ConfigurationTag'

Propietats:

Atributs	Requerit	Descripció
styleId	Sí	Usar el valor 'defaultConfiguration'
i18nService	Sí	Referència al Servei d'Internacionalització

Exemple:

```
<bean id="configurationTag"
class="net.opentrends.openframe.services.web.struts.taglib.configuration.Configura
tionTag">
          <property name="styleId" value="defaultConfiguration"/>
          <property name="i18nService" ref="i18nService"/>
</bean>
```

I a la pàgina JSP (es recomana definir-lo al template de Struts Layout):

```
<fwk:configuration styleId="defaultConfiguration" />
```

2.2.4 Configuració dels tags de Formulari

Configuració del Tag Form

Aquest tag es basa en Struts, pel que sempre podem definir les propietats que el tag 'form' de Struts proporciona. En aquest apartat només es definiran aquelles propietats addicionals que ofereix openFrame.

```
<fwk:form action="accounts.do" reqCode="edit" width="500" method="post">
```

Atributs addicionals:

Atribut	Requerit	Valor
reqCode	Sí	Aquest paràmetre permet especificar quin nom de funció de la Action es cridarà en el moment de fer el submit del formulari. Exemple: Si especifiquem com a 'reqCode=save' es cridarà al mètode 'save' de l'acció de Struts.
layout	Sí	Aquesta propietat permet definir si volem generar o no el layout pel formulari. Per defecte, el tag crea un <table> que agruparà els components input interns. Aquests components interns generaran '<tr>'<td>' de forma automàtica a menys que configurem la seva propietat 'layout' a false. En qualsevol cas, també s'ofereixen tags que ens permetran definir el nombre de columnes i files, crear una fila on ubicar tots els components (cas en el que no crearan <tr>), etc. (veure 'Tags de Layout').
validationProperties	No	Mitjançant aquesta propietat de tipus 'Properties' podem especificar com es farà la validació de les dades entrades al formulari. Veure informació detallada del seu ús a continuació

- Configuració de l'atribut 'validationProperties'

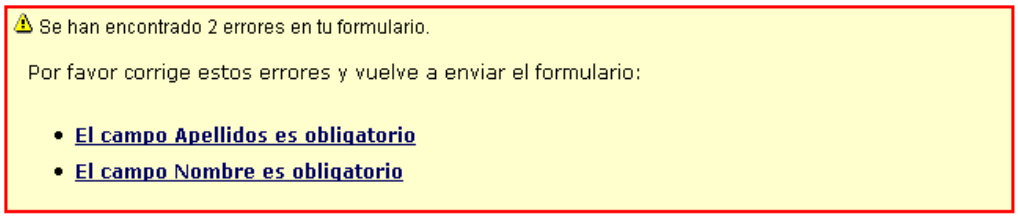
```
<property
name="validationProperties">
      <props>...
</property>
```

Dins el tag <props> podem definir les següents propietats:

Atribut	Requerit	Valor
validationType	Sí	Es permeten els valors: • SERVER. La validació es realitzarà mitjançant Ajax contra el Servei de Validació en el servidor. El resultat de la validació apareixerà a la mateixa pàgina HTML del client • CLIENT. La validació es realitza mitjançant Javascript generat al client
validatorName	Depén validationnType	Especifica quin és el nom de validador definit al fitxer 'validation.xml' del servidor que es farà servir per validar els camps entrats (veure 'Servei de Validació' per més referència) En el cas que s'hagi usat 'SERVER' com a 'validatorType', no cal indicar 'validatorName' si es vol usar com a nom de validador el 'pojoClass' configurat a l'acció associada al formulari.

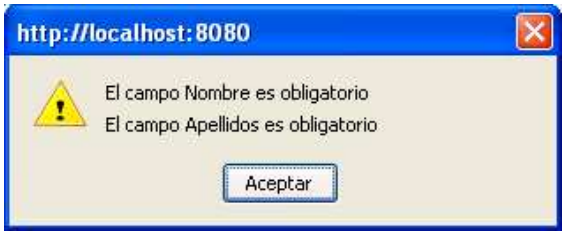
Exemple segons validationType 'SERVER':

```
<property name="validationProperties">
   <props>
          <prop key="validationType">SERVER</prop>
   </props>
</property>
```



Exemple segons validationType 'CLIENT':

```
<bean parent="formTag">
          <property name="styleId" value="actionForm"/>
   <property name="validationProperties">
          <props>
                 <prop key="validationType">CLIENT</prop>
                 <prop key="validatorName">account</prop>
          </props>
          ...
</bean>
```



Configuració de Propietats comunes a tots els tags d'entrada de dades en formulari

Quan desenvolupem una aplicació en la que intervenen formularis que es poden mostrar de manera diferenciada segons els privilegis de l'usuari o l'acció escollida existeixen 2 formes tradicionals de definir-los:

1) Creant una pàgina JSP per cada possible presentació del mateix formulari
2) Imbricant codi condicional que faci el control (segons els privilegis, el paràmetre de la url, etc.) i executi la creació dels tags de diferents maneres

Tant una com l'altra solució requereixen d'un esforç massa gran. En la primera opció, un canvi que impliqui afegir un camp o retocar un ja existent pot requerir de la necessitat de modificar vàries pàgines JSP, mentre que en la segona solució la pàgina es fa molt difícil de llegir i mantenir.

openFrame permet l'ús de modes de formulari que s'aplicaran als components continguts en ell. Segons el mode especificat, els components es mostraran d'una o un altra forma (ocults, editables, no editables, etc.)

Els tags input de openFrame es basen en els tags proporcionats per Struts, de forma que podem sempre definir les seves propietats als nostres fitxers de configuració.

En aquest apartat només es mostren les propietats addicionals que ofereix openFrame:

Propietat	Requerit	Valor
key	Sí	Clau del literal a generar com a 'label' del component.
layout	No	Indicar 'false' si no es volen generar els tags '<tr>','<td>' per als components. Per defecte és 'true' si no indiquem aquesta propietat
accessKey	No	Permet fer ús d'accés per teclat al component. S'ha d'especificar la clau del fitxer d'internacionalització que defineix el caràcter que ha de subratllar (del valor del 'key' definit) i considerar per generar l'event de focus si es fa servir 'ALT+valor key'
mode	Sí	Mode de visualització del component segons el mode que s'hagi assignat al formulari (veure apartat 'Utilització del Servei' per conèixer com assignar el mode del formulari). S'ha d'indicar un valor amb 3 parts separades per ','. La primera

Propietat	Requerit	Valor
		indica en quin mode apareixerà el component si el mode del formulari és de creació, la segona part en quin mode ha d'aparèixer si el mode del formulari és d'edició i la tercera part en quin mode ha d'aparèixer si el mode del formulari és de consulta. Exemple: 'E,E,I' A continuació s'ofereix més detall dels valors possibles i del seu ús.
required	No	Mostrar el camp com obligatori. No cal definir-lo si s'ha definit al 'Servei de Validació' que el camp és obligatori. A continuació s'oferirà més detall dels valors possibles i del seu ús.
tooltipKey, tooltipTitleKey i tooltipOptions	No	Permeten generar un tooltip d'ajuda al component A continuació s'oferirà més detall dels valors possibles i del seu ús.
selectOnFocus	No	En rebre el focus el component tindrà tot el seu contingut seleccionat. Especificar 'false' si no es vol utilitzar aquest comportament per defecte.
i18nService	Sí	Referència al servei d'internacionalització
validationService	Sí	Referència al servei de validació

- **'key' i 'layout '(Generació de etiquetes i layout als components)**

En tota aplicació Web tenim que utilitzar formularis. Aquests solen estar estructurats mitjançant una graella en la que es mostren els components de formulari (camps d'entrada de dades, de selecció, botons d'opció, etc.) i a l'esquerra una etiqueta associada al camp.

Mitjançant Struts, JSTL i l'ús del tag '<label>' (descuidat per molts desenvolupadors però existent a l'estàndard HTML fa molt de temps) la realització d'un component implicaria el següent codi:

```
<tr><td>
   <label for="id" ...>
          <fmt:message key="key.id "/>
   </label>
   <html:text property="id" size="20" maxlength="20"/>
</td></tr>
```

Resulta obvi que a mesura que anem afegint components a la nostra pàgina JSP aquesta es fa feixuga de llegir, ja que incorpora gran quantitat de layout i etiquetes.

Per què no un tag que generi tot aquest codi i fos senzill d'utilitzar? Amb el nou tag proporcionat per openFrame el codi equivalent seria:

```
<fwk:text styleId="id" property="id"/>
```

Exemple:

```
<bean parent="textFieldTag">
   <property name="styleId" value="firstname"/>
   ...
   <property name="key" value="forms.accountForm.field.firstname"/>
```

- **accessKey (accés per teclat)**

Per a poder accedir mitjançant el teclat a qualsevol camp, podem usar l'atribut 'accesskey'. Mitjançant HTML podríem crear el següent codi:

```
<label for="nombre"><u>N</u>ombre</label>
<input type="text" accesskey="n" id="nombre" name="nombre">
```

Mode tradicional

openFrame genera de forma automàtica [2] el codi d'accés als components. En conjunció amb el servei de internacionalització defineix quina és la clau que defineix quin és el caràcter a subratllar.

Exemple:

```
<bean parent="textFieldTag">
   <property name="styleId" value="firstname"/>
   <property name="accesskey"
             value="forms.accountForm.field.firstname.accesskey"/>
```

- **mode**

El mode dels camps permet especificar si un camp "input" ha de ser editable, de lectura, ocult o que no aparegui en el formulari segons el mode de display del

2 Llibreria de generació de access keys: http://www.gerv.net/software/aul/

formulari. Això permet per exemple que un camp que forma part de la clau primària només sigui editable durant la creació, o mostrar camps calculats només quan es visualitza la informació i no quan s'edita.

El comportament per defecte és que els camps es mostrin editables si el mode del formulari es CREATE o EDIT, i es mostrin de lectura (valor text pla sense component HTML i un hidden amb el valor del camp) si el mode del formulari és INSPECT.

Aquest comportament per defecte pot alterar-se en els camps mitjançant l'atribut 'mode'. El valor que s'ha d'assignar a aquest atribut ha de seguir el patró 'X,Y,Z'; on X és el comportament que tindrà el camp en cas que el mode del formulari sigui 'FormUtils.CREATE_MODE', Y el comportament en el mode edició ('FormUtils.EDIT_MODE') i Z en mode de consulta ('FormUtils.INSPECT_MODE'). Els valors possibles per X,Y i Z són:

- E: editable
- I: inspeccionable (de consulta), amb un camp ocult corresponent
- S: mostrar, igual que 'I' però no es genera camp ocult
- N: no es mostra
- P: present (com 'S' si el valor del camp no és null i com 'N' si el valor del camp és nul)
- H: ocult (no es mostra, però sí es genera un camp ocult)
- R : de lectura (es genera el camp però amb l'atribut 'readonly')
- D: desactivat (es genera el camp amb l'atribut 'disabled')

Una vegada especificat en quin mode funcionarà el formulari, els components apareixeran segons el mode especificat, sense que per això s'hagi de generar codi JSP addicional.

- **required**

Es pot especificar que es mostri una imatge o text al principi de l'etiqueta per indicar que un camp és requerit si especifiquem el valor 'true' per la propietat 'required'.

Aquesta propietat **no és necessària** especificar-la si en el Servei de Validació s'ha configurat que el camp és obligatori. El tag accedeix al servei i pregunta si el camp associat és requerit. Per a que això funcioni cal que s'especifiqui la propietat 'validationService' al tag.

El comportament proporcionat per openFrame crea el contingut '
'*' abans del tag HTML '<label>' . Aquest comportament està definit al fitxer 'scripts/ajax/behaviour/openFrame-behaviours.js'. En cas de voler canviar aquest comportament podem accedir al fitxer i fer els canvis necessaris al codi:

Podem configurar l'estil de l'asterisc modificant al fitxer d'estils la següent entrada:

```
label span.required { color: #c32; }
```

- **tooltipKey, tooltipTitleKey y tooltipOptions**

  Aquestes 3 propietats ens permeten especificar l'ajuda contextual als camps.

  - tooltipKey. Clau del literal del missatge a mostrar dins el tooltip
  - tooltipTitleKey. Clau del literal a mostrar si volem mostrar un títol com a capçalera del tooltip
  - tooltipOptions. Opcions aplicades segons la implementació (en el cas actual sota DOM Tooltip).

  Per a ser utilitzat el tag 'Configuration' importa el fitxer js 'scrips/ajax/ajaxtags/openFrame-ajaxtags.js'. Si es vol canviar el comportament per defecte es poden fer els canvis necessaris a aquest fitxer.

  En aquest codi es genera de forma automàtica el link que mostrarà el tooltip en passar per sobre. Addicionalment podem configurar els estils dels tooltips segons la definició dels estils mostrats a continuació:

```
a.tooltip:link {text-decoration: none; }
a.tooltip:visited {text-decoration: none; }
a.tooltip:hover {text-decoration: none; cursor:help; }
a.tooltip:active {}
```

  Un dels estils més útils és 'cursor:help' dins a hover, ja que ens apareixerà el cursor d'ajuda quan passem per sobre de l'etiqueta del component.

NOTA: El cursor d'ajuda no apareix en el gràfic

Configuració del Tag Text

El tag de text permet l'entrada de dades i es basa en el tag de Struts. Per tant, totes les propietats definides a Struts permeten ser definides a openFrame.

A més de les propietats pròpies als components de formulari i de les de Struts s'han incorporat les següents propietats:

Propietat	Requerit	Valor
autoTab	No	Especificar 'true' si es vol realitzar autotabulació en haver introduït el nombre de caràcters especificat a la propietat '**maxLength**' del component.
convertTo	No	Permet definir quines conversions es volen realitzar al valor introduït al camp de forma automàtica en perdre el focus del component. Cal especificar les diferents conversions a aplicar separades per ','. Es proporcionen les següents conversions: • uppercase. Passa tot el contingut a majúscules • lowercase. Passa tot el contingut a minúscules • trim. Elimina els blancs de l'esquerra i dreta del valor. Entre les paraules s'encarrega també de deixar un únic blanc.
showCalendar	No	Indicant 'true' podem especificar que el camp d'edició ha de mostrar un calendari de selecció de data. En cas de que en el servei de validació s'hagi especificat que el camp associat té una validació de data, el tag comprovarà el format definit i serà aquest el que permetrà a openFrame saber quin és el format de la data que ha de copiar en seleccionar una data dins el calendari.

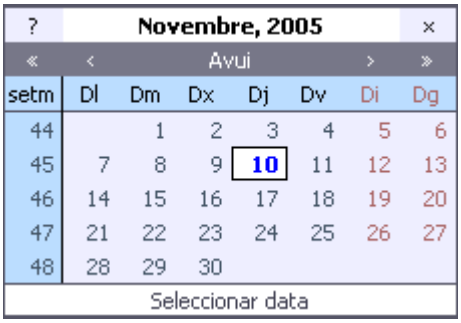
- **showCalendar**

  Per a que el calendari es mostri correctament cal:

  1) Definir el valor 'date' en l'atribut 'depends' dins el fitxer de configuració del 'Servei de Validació'. Definir en 'datePattern' el patró a complir

  2) Incorporar el css a utilitzar

```
<style type="text/css">@import url(css/calendars/dynarch/calendar-
blue.css);</style>
```



Podem usar diferents fitxers d'estils o definir un propi. Els fitxers d'estils existents es troben a 'css/calendars/dynarch'.

Els literals del calendari es troben definits en fitxers '.js' que es troben al directori 'scripts/calendars/dynarch/lang/'. El tag 'Configuration' és qui s'encarrega d'afegir la referència al fitxer corresponent a l'idioma de l'usuari.

Integració amb el Servei de Validació

Per a que s'usi un format de data específic, openFrame consulta per la propietat associada al tag (indicada mitjançant l'atribut 'property' del tag) si es tracta d'un objecte de tipus 'java.util.Date'. Si és així, obté el format que s'ha d'aplicar segons l'idioma de l'usuari. Podem configurar quin serà el format de data utilitzat per cada llenguatge definit una nova clau i el seu patró en la propietat 'localeDatePatternsMap' del fitxer 'property-editors.xml' tal i com es mostra a continuació:

```
...
<bean id="customEditors" class="java.util.HashMap">
<constructor-arg>
   <map>
       <entry key="java.util.Date">
                <bean
class="net.opentrends.openframe.services.i18n.spring.beans.propertyeditors.Cust
omDateEditor">
                      <property name="i18nService" ref="i18nService"/>

                             <property name="localeDatePatternsMap">

<map>
<entry>
<key><value>es</value></key>
<value>dd/MM/yyyy</value>
                             </entry>
                             <entry>
                       <key><value>en</value></key>
                <value>MM/dd/yyyy</value>
      </entry>

               </map>
  </property>
...
```

```
</bean>
```

Configuració del Tag 'Password'

El tag de password permet definir les mateixes propietats que el tag 'Text'.

Exemple:

```
<bean parent="passwordFieldTag">
       <property name="mode" value="E,I,I"/>
       <property name="styleId" value="password"/>
       <property name="key"          value="forms.accountForm.field.password"/>

</bean>
```

Configuració del Tag 'TextArea'

Propietat	Requerit	Valor
rows	Sí	Número de files
cols	Sí	Número de columnes
decoratorProperties	No	Permet especificar la generació d'una àrea amb justificats, formats, ...

- **decoratorProperties**

  Permet especificar la generació d'un textarea en el que es permetin justificats, formats, colors, etc. El contingut aquí introduït serà enviat com HTML.

  Per especificar els decoratorProperties usarem els següents valors:

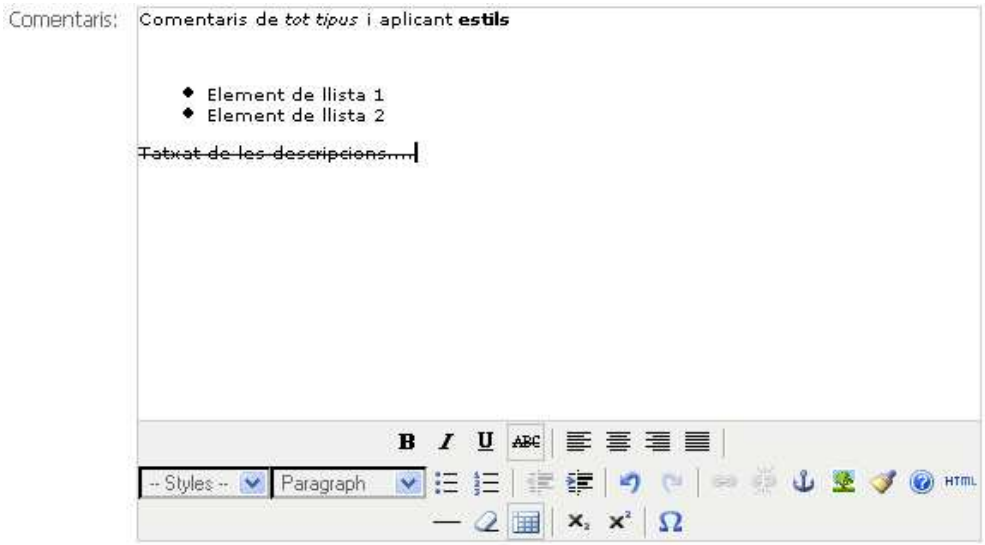
  - theme: 'advanced' o 'simple' (si volem més o menys opcions)
  - mode: 'exact'

  Exemple:

```
<bean parent="textAreaFieldTag">
   <property name="styleId" value="comments"/>                    <property
name="key" value="forms.accountForm.field.comments"/>        <property
name="rows" value="10"/>                                     <property
```

```
name="cols" value="40"/>                                        <property
name="decoratorProperties">
          <map>
                <entry>
                  <key><value>theme</value></key>
                  <value>advanced</value>
                </entry>
                <entry>
                  <key><value>mode</value></key>
                  <value>exact</value>
                </entry>
</map>
   </property>
</bean>
```

Mitjançant l'exemple a dalt mostrat tindríem un component com el mostrat a continuació:



Configuració del Tag 'Select'

Permet definir les mateixes propietats que les definides per Struts i les propietats generals a tots els tags d'entrada de formulari definides anteriorment.

Addicionalment es proporcionen 2 funcionalitats prou importants:

a) Generació d'opcions segons una query de base de dades
b) Generació de les opcions segons el valor escollit en un altre select de forma dinàmica sense refresc de la pàgina

- **Generació de les opcions segons query de la base de dades**

  Si volem mostrar una llista d'opcions que prové d'una base de dades openFrame openFrame ofereix la possibilitat de definir la font de les dades.

  Per això cal que seguim els següents passos:

  1) Definir l'adaptador que s'usarà basat en Hibernate

```
<bean name="defaultOptionBaseHibernateAdapter"
        class="net.mlw.vlh.adapter.hibernate3.Hibernate30Adapter">
        <property name="sessionFactory" ref="sessionFactory"/>
        <property name="defaultNumberPerPage" value="5"/>
        <property name="defaultSortColumn" value="id"/>
        <property name="defaultSortDirection" value="asc"/>
        <property name="removeEmptyStrings" value="true"/>
</bean>
```

  El seu ús és equivalent al definit pel Servei de Llistats.

  2) Gestors de les llistes

```
<bean name="defaultOptionValueListHandler"
        class="net.mlw.vlh.DefaultValueListHandlerImpl">
        <property name="config.adapters">
                <map>
                        <entry key="categoriesList">
                                <bean parent="defaultOptionBaseHibernateAdapter">
                                        <property name="hql">
                                                <value>
                                                        FROM

net.opentrends.openframe.samples.jpetstore.model.Category
                                                        AS vo WHERE 1=1
                                                        /~descn: AND vo.descn LIKE
{descn} ~/
                                                        /~sortColumn: ORDER BY
vo.[sortColumn] [sortDirection]~/
                                                </value>
                                        </property>
                                </bean>
                        </entry>

                </map>
        </property>
</bean>
```

En la propietat 'config.adapters' definirem les diferents queries a executar per a recuperar les llistes. Aquestes queries es troben definides en HQL, pel que obtenim objectes.

- key. Nom de la llista (aquest nom serà utilitzat des de la definició del tag)
- hql. Definició de la consulta Hibernate per a obtenir la llista de valors

```
<entry key="nomLlista">
<bean parent="defaultOptionBaseHibernateAdapter">
       <property name="hql">
       ...
```

3) Definir les següents entrades:

```
<bean id="defaultOptionValueListHandlerHelper"
class="net.mlw.vlh.web.mvc.ValueListHandlerHelper">
<property name="valueListHandler">
       <ref bean="defaultOptionValueListHandler" />
</property>
</bean>


<bean name="defaultOptionListSource"
class="net.opentrends.openframe.services.web.taglib.util.options.vlh.hibernate3
.VlhOptionListSourceImpl">
<property name="valueListHandlerHelper">
       <ref bean="defaultOptionValueListHandlerHelper" />
</property>
</bean>
```

4) Definir la informació de la font de la nostra llista

- optionListName: Referència a la definició de la llista definida al pas 2
- optionLabelName: Atribut de l'objecte recuperat que servirà d'etiqueta de l'opció
- optionLabelProperty: Atribut de l'objecte recuperat que servirà de valor de l'opció

```
<bean id="categoriesOptionListSource" parent="defaultOptionListSource">
              <property name="optionListName" value="categoriesList"/>
              <property name="optionLabelName" value="name"/>
              <property name="optionLabelProperty" value="id"/>
```

```
</bean>
```

5) Incorporar la llista al servei d'opcions

```
<bean id="optionsListService"
class="net.opentrends.openframe.services.web.taglib.util.options.OptionsListSer
viceBase">
<property name="optionsListSources">
        <map>
                <entry key="categoriesList">
                <ref bean="categoriesOptionListSource"/></entry>

                <entry key="animalsList">
                <ref bean="animalsOptionListSource"/></entry>

        </map>
</property>
</bean>
```

6) Per últim definir a la propietat 'optionsListService' del tag una referència al servei d'opcions (propietat 'optionsListService') i afegir el nom de la llista a la propietat 'optionsListName':

```
<bean id="selectFieldTag"
class="net.opentrends.openframe.services.web.struts.taglib.forms.fields.SelectF
ieldTag">
...
        <property name="optionsListService" ref="optionsListService"/>
...
```

```
<bean parent="selectFieldTag">
        <property name="key" value="forms.accountForm.field.preferredCategory"/>
                                <property name="styleId"
value="preferredCategoryId"/>                <property name="optionsListName"
value="categoriesList"/>
</bean>
```

- **Generació de les opcions segons el valor escollit en un altre select**

En cas de voler presentar la llista d'opcions d'un select segons la selecció realitzada a un altra selecció, definirem a més del definit en la secció anterior un nou atribut 'selectFieldSource' en el que definirem:

- source. Nom de la propietat 'styleId' utilitzada en el select font

- paramName. Nom del paràmetre de la query definit entre {}.

Per exemple, imaginem que volem mostrar la llista d'animals segons la categoria escollida.

Podem definir la hql de la llista 'animalsList' tal i com es mostra a continuació:

```
<entry key="animalsList">
<bean parent="defaultOptionBaseHibernateAdapter">
       <property name="hql">
              <value>
              FROM
       net.opentrends.openframe.samples.jpetstore.model.Product
              AS vo WHERE 1=1
              /~category: AND vo.category.id LIKE {category} ~/
              /~sortColumn: ORDER   BY vo.[sortColumn]
[sortDirection]~/
              </value>
       </property>
</bean>
</entry>
```

En aquesta hql apareix el paràmetre {category} dins el LIKE que és el que ens permetrà lligar que se'ns retornin els productes amb la categoria indicada.

Per a que s'envii com a criteri valor dins {category} el valor escollit a la select source definirem com a 'paramName' el valor 'category'.

```
<bean parent="selectFieldTag">
       <property name="key" value="forms.accountForm.field.preferredAnimal"/>
                              <property name="styleId"
value="preferredAnimalId"/>                  <property name="optionsListName"
value="animalsList"/>               <property name="selectFieldSource">
       <map>
             <entry>
               <key><value>source</value></key>
               <value>preferredCategoryId</value>
             </entry>

             <entry>
               <key><value>paramName</value></key>
               <value>category</value>
             </entry>
</map>

       </property>
</bean>
```

Configuració del Tag 'Checkbox'

Permet definir les mateixes propietats que les definides per Struts i les propietats generals a tots els tags d'entrada de formulari definides anteriorment.

Exemple:


```
<bean id="checkboxFieldTag"
class="net.opentrends.openframe.services.web.struts.taglib.forms.fields.CheckboxFi
eldTag">
          <property name="i18nService" ref="i18nService"/>
          <property name="validationService" ref="webValidationService"/>
          <property name="layout" value="true"/>
</bean>
```

```
<bean parent="checkboxFieldTag">
   <property name="styleId" value="lower18"/>
   <property name="key" value="forms.accountForm.field.lower18"/>
</bean>
```

Configuració del Tag 'File'

Aquest tag permet especificar l'ús d'un fitxer de upload

Veure 'Servei de Upload de Fitxers' per a més referència de la configuració necessària.

El tag requereix que es configuri la propietat 'encType' del tag 'Form' amb el següent valor:

"multipart/form-data"

Exemple:

```
<bean parent="fileFieldTag">
          <property name="styleId" value="file"/>
          <property name="mode" value="E,I,I"/>
          <property name="key" value="forms.fileForm.field.file"/>
</bean>
```

2.2.5 Configuració dels tags de Llistats

Veure document 'Servei de Llistats'.

2.3 Configuració dels tags en el JSP

El nou servei de configuració permet definir totes les propietats de text o referències de components en el JSP. Això fa que no sigui necessari tenir definit un styleId que faci referència en el action-services-XXX.xml en la propietats tagsConfiguration de l'action.

Cal remarcar que per propietats complexes, com Maps o Lists, no es dona suport i necessàriament s'han d'escriure en els fitxers d'Spring i relacionar-les amb el styelId.

Aquesta utilitat simplifica el desenvolupament ja que es pot tenir configurat totes les dades del tag en el propi JSP sense necessitat d'haver d'escriure les propietats en fitxers Spring.

Per fer referències a components dins el tag JSP s'ha creat l'atribut *services* en tots el tags que permet especificar la propietat i la referència del component que se li ha d'injectar al tag.

Seguidament es mostra un exemple.

```
<fwk:text styleId="password" layout="false" property="password"
services="i18nService:i18nService,validationService:webValidationService" />
```

En l'exemple anterior s'estan injectant els beans d'spring amb id 'i18nService' i 'webValidationService' en les propietats 'i18nService' i 'validationService' del tag, respectivament.

2.4 Tags avançats

Aquests tags hi són a partir de la versió 1.1.

2.4.1 Dirty Form Warning

El dirty form warning és una funcionalitat que permet a l'usuari avisar quan es córrer el risc de perdre els canvis en un formulari.

Quan es clicka un link i produeix una descàrrega del formulari que té canvis, surt un missatge d'avís de pèrdua de les dades. Llavors l'usuari pot decidir que es procedeixi a la descàrrega del formulari per presentar una nova pàgina o decidir aturar l'acció per no perdre les dades.

L'event que s'escolta per llençar l'avís és el onBeforeUnload suportat per Internet Explorer 6 i FireFox 1.5.

El tag que dona aquesta funcionalitat és el dirtyFormWarning

Dependències:

ajaxtags-1.1.jar
openFrame-dirtyFormWarning-tag.js

Atributs	Requerit	Descripció
styleId	No	Identificador per lligar amb configuració XML
source	Sí	Id del formulari que és vol escoltar dels canvis.
messageKey	No	Clau internacionalitzada del missatge
Message	No	Text del missatge a mostrar

Exemple d'ús:

```
<fwk:dirtyFormWarning  source="actionForm" messageKey="errors.lostChanges"  />
```

2.4.2 Search Panel

El panell de cerca és una utilitat que permet a l'usuari poder trobar el valor adient d'un camp. La cerca és fa mitjançant mostrant un panell amb un camp per omplir dades a cercar i un botó per engegar la cerca. El resultat és mostra sota del camp de text.

El panell de cerca permet mostrar de les dades que compleixen la condició més dades relacionades. Per exemple si s'estan buscant cognoms de persones, el panell de cerca permet mostrar a part del cognom, el nom, el càrrec que ocupa, centre, etc per tal d'ajudar a l'usuari a escollir correctament i en cas de repetició, poder discriminar.

Aquesta és la gran diferència amb l'autocomplete o els selects, que només mostren un dada en concret.

Les cerques es poden fer tantes com es vulguin, però un cop es seleccioni una dada, el panell s'amagarà. Si es torna a pitjar el botó de cercar, el panell de cerca tornarà a aparèixer amb les dades de la última cerca.

El tag que ofereix aquesta funcionalitat és el searchPanel

Dependències:

ajaxtags-1.1.jar
openFrame-ajaxtags-searchPanel.js

Atributs	Requerit	Descripció
styleId	No	Identificador per lligar amb configuració XML
source	Sí	Id del component gràfic que escolarà l'event onClick per mostrar el panell.
popupId	Si	Identificador del div que conté el panell de cerca.
target	Si	Camp de text que es copiarà el valor seleccionat
optionListName	Si	Query que executa el panell.
styleClass	Si	Estil CSS per aplicar al panell.
indicator	Si	Identificador del div que es mostra al fer la petició a servidor.
tableProperties	Si	Nom de les propietats del bean que es volen mostrar en la cerca
selectedProperty	Si	Propietat a copiar al target
columnLabel	Si	Claus dels noms de les propietats internacionalitzades
method	No	Nom del mètode que s'encarregarà de fer la presentació, enlloc de fer la per defecte.
queryParameter	Si	Nom del paràmetre de la query.
services	No	Parelles de nom servei:servei . nom servei és l'atribut que conté el servei, i servei és el nom del servei en spring.

- **popupId**

Identificador del div que conté el panell de cerques. Va bé per poder definir estils .

- **optionsListName**

Nom de la query a executar. Ha d'estar definida en openFrame-services-weblist.xml.

- **queryParameter**

Nom del paràmetre de la query que se li associarà el valor del camp entrat a buscar.

- **tableProperties**

Llista de propietats del bean que retorna la query. Es permeten propietats relacionades. Per exemple si la query retorna elements R i aquests tenen un relació L que conté elements C, es pot posar L.id, que seria el id de l'element C associat.

- **selectedProperty**

Nom de la propietat del bean que retorna la query que es vol copiar al camp de text target. Aquest propietat pot estar dins les tableProperties.

- **columnLabel**

Claus de multillenguatge pel nom de les columnes de la taula de resultats. El nombre de columnLabels i de tableProperties ha de ser igual. En cas de no ser-ho el searchPanel avisa del conflicte.

- **method**

Nom de mètode javascript que sap fer el pintat del resultats. Aquest paràmetre permet a l'usuari canviar l'aspecte de la presentació de resultats. Aquesta funció ha conèixer el format json que retorna servidor per poder presentar les dades.

Exemple:

openFrame-services-web-list.xml

```
    <bean name="valueListHandler"
           class="net.mlw.vlh.DefaultValueListHandlerImpl">
           <property name="config.adapters">
                  <map>
                         <entry key="accountList">
                                <bean parent="baseHibernateAdapter">
                                       <property name="hql">
                                              <value>
                                                     FROM

    net.opentrends.openframe.samples.jpetstore.model.Account
                                                     AS vo WHERE 1=1
                                                     /~firstname: AND
vo.firstname LIKE {firstname} ~/
                                                     /~lastname: AND vo.lastname
LIKE {lastname} ~/
                                                     /~sortColumn: ORDER BY
vo.[sortColumn] [sortDirection]~/
                                              </value>
                                       </property>
```

```
                                </bean>
                            </entry>
                    </map>
            </property>
    </bean>
```

JSP:

```
<input type="button" value="search" id="searchPanelButton" />
                                                    <fwk:searchPanel

    styleId="searchPanel"

    source="searchPanelButton"
                                                            target="city"

    popupId="searchPanelDiv"

    optionsListName="accountList"

    tableProperties="firstname,lastname,city"

    selectedProperty="city"

    columnLabels="name.label,lastname.label,city.label"

    indicator="indicator"

    queryParameter="firstname" />
```

Per fer servir el searchPanel s'han d'importar els següents scripts:

```
<!--  Search panel -->
<script
   src="/openFrame-samples-jPetstore/scripts/ajax/ajaxtags/openFrame-ajaxtags-
searchPanel.js"
   type="text/javascript">
</script>
<script type='text/javascript'
   src='/openFrame-samples-
jPetstore/AppJava/dwr/interface/searchPanelService.js'></script>
```

2.4.3 Autocomplete

L'autocomplete permet a l'usuari mostrar una llista de suggeriments pel valor que ha escrit. L'usuari clickant una de les opcions mostrades li evita escriure i li corrobora que la dada és correcte.

En el framework aquesta funcionalitat s'obté amb el tag autocomplete. Aquest autocomplete és importat de la llibreria ajaxtags 1.1 i s'ha adaptat perquè cridi al servei de cerques del framework i mostri correctament el resultat. Entre el servidor i client viatja una cadena json.

Dependències:

ajaxtags-1.1.jar
openFrame-ajaxtags-autocomplete.js

Atributs	Requerit	Descripció
source	Sí	Id del component gràfic que escolarà l'event onKeyUp per mostrar els suggeriments.
target	Si	Camp de text que es copiarà el valor seleccionat
optionListName	Si	Query que executa el panell.
className	Si	Estil CSS per aplicar al panell.
indicator	No	Identificador del div que es mostra al fer la petició a servidor.
minimumCharacters	Si	Nombre de caràcters mínims per executar la query.
errorFunction	No	Funció a cridar en cas d'error
emptyFunction	No	Funció a cridar en cas de no haver resultats
postFunction	No	Funció a cridar després de rebre dades.

- **source**

Component gràfic que es vol escoltar els events de teclat per tal de mostrar un autocomplete.

- **target**

Camp de text que es desitja copiar el valor seleccionat en l'autocomplete. Normalment serà igual que el source.

- **optionsListName**

Consulta definida en openFrame-services-web-options-list.xml

- **minimumCharacters**

Nombre de caràcters mínims en el camp de text origen per tal de disparar la consulta.

- **errorFunction**

Funció a executar en cas d'error de dades o de comunicació.

- **emptyFunction**

Funció a executar en cas d'element buit.

- **postFunction**

Funció a executar després de presentar les dades.

Exemple:

openFrame-services-web-options-list.xml

```
<bean name="defaultOptionValueListHandler"
      class="net.mlw.vlh.DefaultValueListHandlerImpl">
    <property name="config.adapters">
   <map>
       <entry key="animalsList">
          <bean parent="defaultOptionBaseHibernateAdapter">
              <property name="hql">
                 <value>
                 FROM
                 net.opentrends.openframe.samples.jpetstore.model.Product
                 AS vo WHERE 1=1
                 /~name: AND vo.name LIKE {name} ~/
                 /~category: AND vo.category.id LIKE {category} ~/
```

```
                    /~sortColumn: ORDER   BY vo.[sortColumn] [sortDirection]~/
                    </value>
                </property>
            </bean>
        </entry>
         </map>
       </property>
</bean>

<bean id="animalsOptionListSource" parent="defaultOptionListSource">
          <property name="optionListName" value="animalsList"/>
                <property name="optionLabelName" value="name"/>
                <property name="optionLabelProperty" value="id"/>
</bean>

<bean id="optionsListService"
class="net.opentrends.openframe.services.web.taglib.util.options.OptionsListServic
eBase">
   <property name="optionsListSources">
       <map>
          <entry key="animalsList"><ref bean="animalsOptionListSource"/></entry>
        </map>
   </property>
</bean>
```

JSP:

```
<fwk:text styleId="preferredProduct2.id" styleClass="fieldtext"
property="preferredProduct2.id" />

<fwk:autocomplete
   source="preferredProduct2.id" target="preferredProduct2.id"
   parameters="preferredProduct2.id" className="autocomplete"
   minimumCharacters="1" indicator="indicator"
   optionsListName="animalsList" />
```

2.4.4 Botons amb imatges

El tag fwk:button permet definir botons amb el contingut que és vulgui, entre d'altres per exemple amb taules, imatges, text, etc.

Aquest tag és basat amb l'element d'html button, que és nou en l'html 4.0.
La missió del tag és encapsular aquest element, oferint tots els serveis que dona l'especificació.

Atributs	Requerit	Descripció
styleId	Sí	Id del botó
styleClass	No	CSS del botó
Tabindex	No	Índex de tabulació
Value	No	Camp valor
Title	No	Títol del botó
Lang	No	
Dir		
Disabled	No	Si està deshabilitat o no

Events:
onblur,onchange,onclick,ondbclick,onfocus,onkeydown,onkeypress,onkeyup,onmousedown,onmousemove,onmouseout,onmouseover,onmouseup

Exemple:

```
<fwk:button styleId="desa" type="submit">
   <img src="<c:url value="/images/save.jpg"/>" />
   <bean:message key="jsp.includes.submit" />
</fwk:button>
```

2.4.5 Tag Swap Select

Aquest component ofereix un tag que permeten moure o copiar elements entre dos selects múltiples. El seu ús es ben senzill.

Dependències:

ajaxtags-1.1.jar
openFrame-options-tag.js

Atributs	Requerit	Descripció
source	Sí	Id del component gràfic que escoltarà l'event onClick per fer l'acció configurada.
action	Si	Acció, que pot ser MOVE o COPY
scope	Si	Àmbit de l'acció. Pot ser ALL o SELECTED.
fromSelect	Si	Select origen
toSelect	No	Select destí.

- **source**

Component gràfic que se li associa a l'event de onClick l'acció configurada. Normalment serà un botó.

- **action**

La acció pot ser moure o copiar elements d'un select cap a un altre. Els possibles valors són COPY o MOVE.

- **scope**

Dades del select origen. Poden ser les dades seleccionades o totes. Els dos possibles valors són ALL o SELECTED

- **fromSelect**

Select origen de dades.

- **toSelect**

Select destí de dades.

Exemple:

```
<!-- Selects -->
<fwk:select styleId="allAnimals" roperty="allAnimals" multiple="true"
styleClass="multipleSelect" />
<fwk:select styleId="hateAnimals" property="hateAnimals" multiple="true"
            styleClass="multipleSelect" />
```

```
<!-- Botons -->
<input type="button" id="moveAllRight" value="&gt;&gt;" lass="botonsswap" />
<input type="button" id="moveRight" value="&gt;" class="botonsswap" />
<input type="button" id="moveLeft" value="&lt;" class="botonsswap" />
<input type="button" id="moveAllLeft" value="&lt;&lt;" class="botonsswap" />

<! -- decoradors de swap -->
<fwk:swapTag source="moveAllRight" scope="ALL" action="MOVE"
             fromSelect="allAnimals" toSelect="hateAnimals" />
<fwk:swapTag source="moveRight" scope="SELECTED" action="MOVE"
             fromSelect="allAnimals" toSelect="hateAnimals" />
<fwk:swapTag source="moveAllLeft" scope="ALL" action="MOVE"
             fromSelect="hateAnimals" toSelect="allAnimals" />
<fwk:swapTag source="moveLeft" scope="SELECTED" action="MOVE"
             fromSelect="hateAnimals" toSelect="allAnimals" />
```

L'exemple és la construcció de dos selects, amb quatre botons que permeten:
- moveAllRight: mou totes les dades del select allAnimals al hateAnimals.
- moveRight: mou les dades seleccionades del select allAnimals al hateAnimals.
- moveAllLeft: mou totes les dades del select hateAnimals al allAnimals.
- moveLeft: mou les dades seleccionades del select hateAnimals al allAnimals.

2.4.6 Pestanyes

Aquest component està compost per dos tags que permeten la construcció de pestanyes (Tabs).

Dependències:

ajaxtags-1.1.jar
openFrame-ajaxtags-tabs.js

Els tags són els següents:

1.1.2.1 TabPanelTag

El tag TabPanelTag és el tag pare que s'encarrega de crear les pestanyes i divs ocults, on es guardarà el contingut de les pàgines de les diferents pestanyes. En el cos d'aquest tag és on definirem totes les pestanyes (TabPageTag).

Les propietats d'aquest tag són les següents:

Propietat	Requerit	Valor
panelStyleId	Si	Id del panell que contindrà totes les pestanyes.
contentStyleId	Si	Part comú de l'id dels divs de les pestanyes.
currentStyleId	Si	Id de la pestanya seleccionada.

- **panelStyleId**

Aquesta propietat defineix l'id del panell pare contenidor dels divs de les pestanyes. Amb aquest id podrem canviar l'estil general dels tabs.

- **contentStyleId**

Part comú de l'id dels divs de les pestanyes. El tag pare crearà cada div amb un id diferent. Aquest id es formarà amb la propietat 'contentStyleId' + '_tab' + index. Aquest índex ve donat pel número de pestanya dins del tag pare (TabPanelTag).
Si per exemple donem a 'contentStyleId' el valor 'content' i definim 3 pestanyes, es crearan 3 divs amb els següents ids:
· content_tab0
· content_tab1
· content_tab2

- **currentStyleId**

Aquesta propietat defineix l'id del la pestanya que està activa (la seleccionada). Amb aquest id podrem definir l'estil de la pestanya activa.

Exemple:

```
<fwk:tabPanel panelStyleId="pestanas"
                contentStyleId="tabContent" currentStyleId="active">
```

1.1.2.2 TabPageTag

El tag TabPageTag defineix una pàgina de les pestanyes. Aquest tag no té sentit si no es troba dins d'un tag TabPanelTag.
Les pestanyes poden ser, bàsicament, de dos tipus: client o servidor.
·Les clients ja tenen el contingut en els divs quan es carrega la pàgina.
· Les servidor tenen els divs buits, es carreguen mitjançant AJAX quan es selecciona la pestanya. Quan això succeeix, s'envia una petició al servidor i es carrega en contingut del div mitjançant Javascript i XmlHttpRequest, sense necessitat de recarregar la pàgina sencera.
En Internet Explorer, no es suporta que un formulari contingui un altra formulari dins. Llavors si la pestanya servidor conté un formulari i la plana actual ja en té un de general, donarà un error.

Les propietats d'aquest tag són les següents:

Propietat	Requerit	Valor
linkStyleClass	No	Classe de la pestanya (per aplicar estils a la pestanya).
isActive	No	Indica si la pestanya és 'enabled' o no (disabled);
existingDivId	No	Id del div on es troba el contingut de la pestanya.
caption	No	Títol de la pestanya.
captionKey	No	Clau que correspon al títol de la pestanya.
defaultTab	No	Indica el tab actiu per defecte.
imageCaption	No	Indica la ruta de la imatge que volem afegir a la pestanya.
imagePosition	No	Indica posició de la imatge que volem afegir a la pestanya.
baseUrl	No	Indica la url on trobem el contingut de la pestanya.
parameters	No	Paràmetres de la url.

- **isActive**

Si el valor es 'false', la pestanya quedarà desactivada (disabled) i no es podrà seleccionar.

- **existingDivId**

Id del div on es troba el contingut de la pestanya. Això s'utilitza quan el contingut de la pestanya no es vol posar dins del cos (body) del tag de la pestanya (TabPageTag), sinó que es vol usar el contingut d'un altre div.

- **defaultTab**

Si el valor es 'true', la pestanya quedarà activada al carregar la pàgina. Només es pot definir una pestanya com a 'defaultTab'.

- **imageCaption**

Afegeix una imatge a la pestanya. Indica la ruta de la imatge que volem afegir. Per defecte posa la imatge a l'esquerra del títol(caption o captionKey) de la pestanya.

- **imagePosition**

Indica la posició de la imatge. Els possibles valors són:
· left
· right
Per defecte és 'left'.

- **baseUrl**

Només per pestanyes carregades mitjançant servidor. Indica la url on trobem el contingut de la pestanya. Aquesta url ha de retornar una jsp, que serà el contingut.

Exemple pestanya client:

```
<fwk:tab captionKey="tabs.account.pestanya2"
                        styleClass="contingut_tab" linkStyleClass="pestanya2"
imageCaption="/openFrame-samples-jPetstore/images/ast.gif"
                        imagePosition="right">
```

Exemple pestanya servidor:

```
<fwk:tab captionKey="tabs.account.server"
          styleClass="contingut_tab"
          linkStyleClass="pestanya4"
baseUrl="accountList.do?reqCode=addRowEditList"/>
```

Exemple complet:

```
<fwk:tabPanel panelStyleId="pestanas"
```

```
    contentStyleId="tabContent" currentStyleId="active">
    <fwk:tab captionKey="tabs.account.pestanya1" defaultTab="true"
            styleClass="contingut_tab" linkStyleClass="pestanya1"
            imageCaption="/openFrame-samples-jPetstore/images/asc.gif">
            <fwk:text styleId="firstname" styleClass="fieldtext"
                                property="firstname">
    </fwk:tab>
    <fwk:tab captionKey="tabs.account.pestanya2"
            styleClass="contingut_tab" linkStyleClass="pestanya2"
imageCaption="/openFrame-samples-jPetstore/images/ast.gif"
            imagePosition="right">
            <fwk:text styleId="country" property="country">
    </fwk:tab>
    <fwk:tab captionKey="tabs.account.server"
            styleClass="contingut_tab" linkStyleClass="pestanya4"
baseUrl="accountList.do?reqCode=addRowEditList"/>
</fwk:tabPanel>
```

2.4.7 GridBagLayout

El tag de GridBagLayout és un component que permet la creació de taules que ofereixen un fàcil control de les propietats dels elements que continguin. Mitjançant la definició d'aquest Tag podrem controlar les propietats de cada cel·la (TD) individualment o conjuntament (amb les propietats de cada línia –TR- o de tota la taula -TABLE-).

Per definir les propietats pròpies a la Taula, als TDs i als TRs, s'han de definir el el tag les propietats pròpies de les Taules, dels TDs i dels TRs:

Les propietats referents al component taula es definiran amb el seu nom i el prefix 'grid'. Així, si volem definir la propietat 'style' de la taula, definirem la propietat gridStyle en el tag:

```
gridStyle="border:1px solid black;"
```

Les propietats referents als components de la taula (TD i TR) es definiran amb el seu nom habitual. Per especificar a quina o quines cel·les es vol aplicar la propietat, es farà de la següent manera:

Si ens volem referir a una cel·la en concret, indicarem el número de fila i de columna que ocupa, separat pel caràcter ':'. Afegirem el valor de la propietat després el caràcter '=':

```
style="2:1=border:1px solid black"
```

Així, segons aquest exemple, la cel·la de la fila 2 columna 1 tindrà l'estil 'border:1px solid black'.

Si volem definir una propietat per més d'una cel·la, ho farem separant les definicions amb el caràcter ',':

```
style="2:1=border:1px solid black,2:2=background-color:blue"
```

Si volem definir una propietat per tota una fila o tota una columna, ho farem mitjançant el caràcter '*':

```
style="2:*=border:1px solid black "
```

En l'exemple anterior, l'estil definit s'aplicarà a tota la fila 2. En aquest cas, però, s'aplicarà l'estil a totes les cel·les (TDs) de la fila, no a l'element fila (TR). Si volem aplicar la propietat al TR, ho farem mitjançant '**':

```
style="2:**=border:1px solid black"
```

En aquest cas, l'estil s'aplicarà al TR de la fila 2. Això només ho podrem fer amb les files, ja que compten amb l'element TR. Per les columnes, només podrem aplicar '*', i l'estil s'aplicarà a tots els TDs de la columna individualment.

També podem definir alguna propietat per totes les cel·les de la taula mitjançant '*:*':

```
style="*:*=background-color:blue"
```

En aquest exemple s'aplica l'estil a tots els TDs de la taula. Si volem fer-ho amb tots els TRs, ho farem mitjançant '*:**':

```
style="*:**=background-color:blue"
```

La següent taula recull el diferents valors que hem definit per referir-nos a les cel.les:

Valor	Definició
1:1	Fila 1 columna 1
*:1	Tots els TDs de la columna 1
1:*	Tots els TDs de la fila 1
1:**	TR de la fila 1
:	Tots els TDs de la taula
*:**	Tots els TRs de la taula

A més de les propietats pròpies a les Taules,TDs i TRs s'han incorporat les següents propietats:

Propietat	Requerit	Valor
size	Si	Defineix el nombre de files i columnes que tindrà el grid: Exemple: size="2,3" -> Crearà 2 files i 3 columnes.

El contingut de cada cel·la del grid el definirem en el cos del Tag GridBadLayout. Cada node que trobem serà el contingut de cada cel·la, tenint en compte l'ordre en que els anem trobant. Si volem que una cel·la contingui més d'un element, els haurem d'agrupar algun altre element

(com, per exemple, un 'div' o un 'span'), ja que el tag parssejarà el seu contingut com si es tractés d'un document XML, posant en cada cel·la del grid el contingut de cada node de primer nivell:

```
<fwk:gridBagLayout size="1,3"
                                gridBorder="1"
                                gridStyle="border:1px solid black;"
                                colspan="1:1=2"
                                rowspan="1:1=2">

   <span><input type="text" name="text1" id="text1" value="text1"/></span>
   <input type="text" name="gridText" id="gridText"/>
   <span><input type="text" name="gridT" id="gridT" value="gridT"/></span>

</fwk:gridBagLayout>
```

En l'exemple anterior, es crearà una taula amb una fila i 3 columnes. En la primera cel·la hi posarà el contingut del primer node del cos del tag:

```
<span><input type="text" name="text1" id="text1" value="text1"/></span>
```

En la segona hi posarà el segon node:

```
<input type="text" name="gridText" id="gridText"/>
```

I, finalment, en la tercera, hi posarà el tercer i últim node:

```
<span><input type="text" name="gridT" id="gridT" value="gridT"/></span>
```

· Errors de construcció del grid

Si en el contingut del tag hi tenim més elements que cel·les hem definit en l'atribut 'size', el tag llançarà una excepció. Si, en canvi, hi ha més cel·les que elements, el tag crearà les cel·les restants buides.

El tag suporta també les propietats 'colspan' i 'rowspan'. Aquestes propietats es defineixen quan una cel·la ha d'ocupar més una fila o d'una columna. Amb aquestes propietats es poden

definir incongruències. En aquests casos, el tag llançarà excepcions. Aquestes incongruències poden ser:

·Definir una propietat fora de límits:
Això succeeix quan definim una propietat 'colspan' o 'rowspan' amb un valor que està fora dels límits del grid:

```
<fwk:gridBagLayout size="1,3" colspan="1:3=2">
  ...
```

En l'exemple anterior, definim la propietat 'colspan' de la tercera i última columna cel·la del grid amb valor 2. Això només podria ser si el grid tingués 4 columnes. Així, el tag llançarà una excepció indicant el motiu de l'error.

·Definir una propietat d'una cel·la que està solapada per la propietat 'colspan' o 'rowspan' d'una altra cel·la:

```
<fwk:gridBagLayout size="1,3" colspan="1:1=2"
                              style="1:2=border:1px solid black">
  ...
```

En l'exemple anterior definim la propietat 'style' de la cel·la 1:2, però aquesta cel·la està solapada per la propietat 'colspan' de la cel.la 1:1. Així, el tag llançarà una excepció indicant l'error.

·Exemple complet de GridBagLayout

```
<fwk:gridBagLayout size="3,3"
                                gridBorder="1"
                                gridStyle="border:1px solid black;"
                                colspan="1:1=2"
                                rowspan="1:1=2"
                                style="*:*=border:1px solid
                                       black;,*:0=background-color:blue"
                                onmouseover="*:**=this.className='selected';"
                                onmouseout="*:**=this.className='zebra0';">

   <span><input type="text" name="span1" id="span1" value="span1"/></span>
   <input type="text" name="gridText" id="gridText"/>
   <span><input type="text" name="gridT" id="gridT" value=" gridT "/></span>

   <span><input type="text" name="span2" id="span2" value="span2"/></span>
   <input type="text" name="gridText" id="gridText" value="gridText"/>
   <span><input type="text" name="gridT" id="gridT" value="gridT"/></span>
```

```
    </fwk:gridBagLayout>
```

2.4.8 Paginació de selects

El pagedSelect permet a l'usuari mostrar una llista paginada de suggeriments pel valor que ha escrit. Aquest només mostra un camp en concret, que és el que és copia al camp de text.
L'usuari clickant una de les opcions mostrades li evita escriure i li corrobora que la dada és correcte.

És molt útil per mostrar llistes suggeriments de molts elements, ja que té la paginació.

Aquest pagedSelect és importat de la llibreria ajaxtags 1.1 i s'ha adaptat perquè cridi al servei de cerques del framework i mostri correctament el resultat. Entre el servidor i client viatja una cadena json.

Dependències:

ajaxtags-1.1.jar
ajaxtags-openFrame-pagedSelect.js

Atributs	Requerit	Descripció
styleId	Sí	Id del component gràfic.
selectedKey	Si	Camp de la query que serà la clau de la option
selectedValue	Si	Camp de la query que serà el valor de la option
property	Si	Nom del component.
dependentFields	No	Camps amb el valor dels queryParameters
queryParameters	Si	Paràmetres de la query
optionsListName	Si	Nom de la query

- **styleId**

Id del component gràfic. Serveix per poder aplicar estils al component i poder-lo configurar a partir d'un xml (action-servlet-xxx.xml).

- **selectedKey**

Camp de la query que serà la clau de la option

- **selectedValue**

Camp de la query que serà el valor de la option

- **property**

Nom del component.

- **dependentFields**

Valor que tindran els paràmetres
Aquest camp contindrà tots els valors dels paràmetres de la query (queryParameters) separats per comes.

- **queryParameters**

Nom dels paràmetres dins la query. En la query es fa referència a aquests noms mitjançant la nomenclatura:

/~nomParametre: ... ~/

Aquest camp contindrà tots els paràmetres de la query separats per comes.

- **optionsListName**

Nom de la query. Aquesta query estarà definida dins de "openFrame-services-web -list.xml".

Exemple:

openFrame-services-web-list.xml

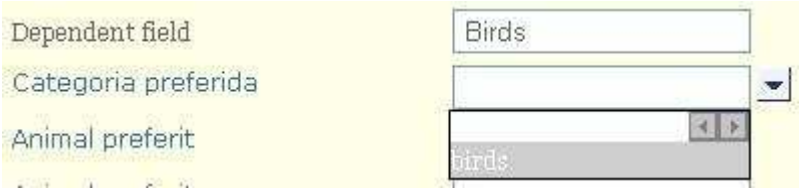
```
    <bean name="valueListHandler"
            class="net.mlw.vlh.DefaultValueListHandlerImpl">
            <property name="config.adapters">
                    <map>
                            <entry key="categoriesList">
                                    <bean parent="baseHibernateAdapter">
                                            <property name="hql">
                                                    <value>
FROM
net.opentrends.openframe.samples.jpetstore.model.Category AS vo

WHERE 1=1
   /~descn: AND vo.descn LIKE {descn} ~/
   /~sortColumn: ORDER BY vo.[sortColumn] [sortDirection]~/
                                                    </value>
```

```
                                        </property>
                                    </bean>
                                </entry>
                            </map>
                        </property>
                    </bean>
```

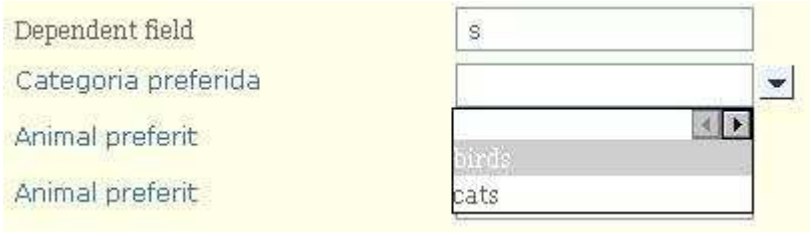
JSP:

```
    <input type="text" name="depValue" value="Birds"/>

    <fwk:pagedSelect
     styleId="preferredCategory.id"
     selectedKey="id"
     selectedValue="name"
     property="preferredCategory.id"
     dependentFields="depValue"
     queryParameters="descn"
     optionsListName="categoriesList"/>
```

Exemple amb un sol resultat:



Exemple amb varis resultats (amb paginació):



2.4.9 Format i limitació d'entrada de dades

Aquest component permet controlar el format i la limitació de l'entrada de dades d'un camp d'un formulari. Aquest component el podem utilitzar de dues maneres: mitjançant un decorador (**fwk:formatKeyStroke**) o directament en la configuració del tag text del Framework (**fwk:text**).

És molt habitual el seu ús en números de telèfon, targetes de crèdit, dates, etc.

Dependències:

masks.js

Atributs	Requerit	Descripció
mask	Sí	Màscara que es vol aplicar a un camp d'un formulari.
maskType	No	Tipus de màscara que es vol aplicar: text, date, number
source	*	Nom del component.

- **mask**

Màscara que volem aplicar a un camp d'un formulari. Depenent del tipus de màscara seguirem un patró o un altre.

- **maskType**

Tipus de màscara. Pot ser de 3 tipus:

· Text: màscara de text.
Amb aquest tipus de màscara podem aplicar els següents patrons:

x: lletra [a-z].
X: lletra [A-Z].
#: número [0-9].
*: lletra o número [a-z][A-Z][0-9].

Qualsevol altre caràcter que aparegui a la màscara s'escriurà automàticament.

Exemple:

xxxx## -> hola22

(###) #x*-#### -> (614) 7a7-6094
telf (##) ###.##.## -> telf (93) 311.34.89

· Date: màscara de data.
Amb aquest tipus de màscara podem aplicar els següents patrons:

`d:` dia [0-9].
`m:` mes [0-9].
`y:` any [0-9].

Exemples:

dd/mm/yyyy -> 24/05/1980
m/dd/yyyy -> 2/05/1980
yyyy.mm.dd -> 1972.02.28

· Number: màscara numèrica.

`$:` indica la moneda (apareix automàticament).
`#:` número [0-9].
`0:` número 0. S'utilitza pels decimals (apareix automàticament).
`+:` Iindicador número positiu (apareix automàticament).
`−:` Iindicador número negatiu (apareix automàticament).
`,:` Separador dels milers, milions, etc.. (apareix automàticament).
`.:` Indica on comencen els decimals.
`(:` principi parèntesis (apareix automàticament).
`):` fi parèntesis (apareix automàticament).

Exemples:

0#####.## -> 000534.23
+##,###.## -> +534.23
(-#,###.##) -> (-534.23)

* Per defecte el tipus de màscara és "text".

- **source**

Id del camp al qual volem aplicar la màscara. Només aplicable al decorador formatKeyStroke, ja que en el tag Text el camp al qual es vol aplicar la màscara és ell mateix.

Exemple amb decorador (fwk:formatKeyStroke):

```
<fwk:text
 styleId="firstname"
 property="firstname"/>

<fwk:formatKeyStroke
 source="firstname"
 mask="xxxx##"/>
```

Exemple sense decorador (fwk:text):

```
<fwk:text
 styleId="firstname"
 property="firstname"
 mask="xxxx##"/>
```

2.5 Llistats editables

Els llistats editables es basen en el suport de les propietats indexades. Per poder donar aquest suport s'han construït un seguit de tags per tal que es pugui fer un gestió correcte d'aquesta informació.

2.5.1 Introducció

Una propietat indexada és:

```
<fwk:text styleId="city" property="direccions[0].ciutat" />
```

Aquest camp mostra de la llista de direccions del bean pare, la ciutat de la primera direcció.

En la versió 1.0 openFrame sap renderitzar aquesta propietat, però no sap construir aquesta llista de direccions en la baixada del submit dins el bean pare.

El problema principal que hi ha per construir la col·lecció de direccions, és que ningú coneix quins objectes ha de tenir dins. Per poder informar a openFrame d'aquesta tasca s'ha construït

un tag anomenat **DefinePropertyTag**. Tota llista del bean que és vol materialitzar en la baixa és necessari que estigui definida amb aquest tag.

Les col·leccions editables poden provenir de moltes fonts.

Un cas típic es que en el load del bean del formulari es carreguin també seves col·leccions per després mostrar en pantalla.

Un altra cas seria aquell on les col·leccions tenen molts elements. Aquí és molt aconsellable fer servir la ValueList ja que dona suport a la paginació, i navegació entre pàgines de manera amigable. La gran diferència amb el model primer, es que aquesta llista es carrega mitjançant una query definida en openFrame-web-lists.xml. Per tal que els tags de propietats indexades es renderitzin correctament, s'ha de enganxar-li al bean del formulari la col·lecció que ha vingut de la valueList. Aquest enganxament es pot fer amb el **SetListTag**, on se li ha d'indicar quina és la propietat indexada del bean del form, i on és la llista ha afegir.

El suport a llistats editables també ofereix el poder afegir, seleccionar i esborrar elements d'un llistat. Aquests pressuposen que el llistat està contingut dins una table de HTML.

2.5.2 SetListTag

El tag SetListTag és un component que permet omplir els elements d'una llista del Bean d'un formulari amb els d'una llista que tenim guardada en qualsevol scope de la pàgina. Per exemple, si les dades provenen d'una valuelist, haurem d'omplir la llista del Form Bean amb la propietat 'list' de l'scope 'page':

```
<fwk:setList listProperty="accounts" scopeKey="list" scope="page"/>
```

Aquest exemple mostra com agafar la llista provinent de la ValueList i l'afegeix a la propietat accounts del bean del formulari.

Les propietats d'aquest tag són les següents:

Propietat	Requerit	Valor
listProperty	Si	Nom de la llista en el Form Bean.
scopeKey	Si	Clau que té la llista en l'scope seleccionat.
scope	Si	Scope on es troba la llista amb la clau definida a 'scopeKey'.

- **listProperty**

Nom de la propietat indexada en el bean del formulari.

- **scopeKey**

Nom de la clau que conté la llista a afegir al bean del formulari.

- **scope**

Aquesta propietat només accepta 3 possibles valors:
· page
· request
· session

2.5.3 DefinePropertyTag

El tag DefinePropertyTag serveix per informar a openFrame quins tipus de classe conté la col.lecció definir la classe que tenen els objectes continguts en la llista del Form Bean. Això es fa per poder fer el 'binding' dels nous objectes que es creïn a l'editar el llistat amb el Form Bean.

Les propietats d'aquest tag són les següents:

Propietat	Requerit	Valor
listProperty	Si	Nom de la llista del bean del formulari.
itemsClass	Si	Classe dels objectes continguts en la llista del bena del formulari.

```
    <fwk:defineProperty listProperty="accounts"
itemsClass="net.opentrends.openframe.samples.jpetstore.model.Account"/>
```

Aquest tag crearà dos inputs ocults amb el següent format:

<input type="hidden" name="__Metadata.nom_llista. itemsClass__" value="classe_items_llista"/>
<input type="hidden" name="__Metadata.nom_llista. lastIndex__" value="tamany_llista"/>

Exemple:

```
<input type="hidden" name="__Metadata.accounts.itemsClass__"
value="net.openframe.samples.model.Account"/>
<input type="hidden" name="__Metadata.accounts.lastIndex__" value="2"/>
```

2.5.4 SelectionTag

El tag SelectionTag serveix per seleccionar i deseleccionar tots els elements d'un llistat editable.

Les propietats d'aquest tag són les següents:

Propietat	Requerit	Valor
listName	Si	Nom de la llista en el Form Bean.
selectionProperty	Si	Nom de la propietat de selecció.
source	Si	Nom de l'element disparador de l'event.
target	Si	Id de la taula que conté el llista editable
action	Si	Acció de selecció o deselecció.

- **selectionProperty**

Aquesta propietat defineix el camp del Form Bean que indicarà si el registre al qual pertany està seleccionat. El camp ha de ser un valor booleà i es representarà amb un element 'checkbox'.

- **source**

Aquesta propietat defineix l'id de l'element al qual s'hi s'assignarà l'event 'onclick' que dispararà la funció de seleccionar o deseleccionar tots els 'checkbox' de la taula.

- **target**

Els llistats editables es faràn sobre taules. La taula que contingui els registres haurà de tenir un id. Aquest id és el que hem d'indicar en la propietat 'target'.

- **selectionProperty**

Acció que s'ha d'executar. Els possibles valors són:
· selectAll (seleccionar tots els registres)

· unselectAll (deseleccionar tots els registres)

Un exemple de la seva utilització podria ésser el següent:

```
    <fwk:selection source="selectionButton" target="taulaId" listName="accounts"
selectionProperty="selected" action="selectAll"/>
    <fwk:selection source="unselectionButton" target="taulaId" listName="accounts"
selectionProperty="selected" action="unselectAll"/>
```

En l'exemple anterior s'assigna a 2 botons (selectionButton i unselectionButton) les funcions de seleccionar i deseleccionar tots els registres de la taula (amb id 'taulaId') respectivament.

2.5.5 DeleteRowsTag

El tag DeleteRowsTag serveix per eliminar els elements seleccionats d'un llistat editable.

Les propietats d'aquest tag són les següents:

Propietat	Requerit	Valor
listName	Si	Nom de la llista en el Form Bean.
selectionProperty	Si	Nom de la propietat de selecció.
source	Si	Nom de l'element disparador de l'event.
target	Si	Id de la taula que conté el llista editable.

- **selectionProperty**

Aquesta propietat defineix el camp del Form Bean que indicarà si el registre al qual pertany està seleccionat. El camp ha de ser un valor booleà i es representarà amb un element 'checkbox'. Tots els 'checkbox' seleccionats indicaran els registres a esborrar.

- **source**

Aquesta propietat defineix l'id de l'element al qual s'hi s'assignarà l'event 'onclick' que dispararà la funció de esborrar tots els registres seleccionats.

- **target**

Id de la taula on s'esborraran els registres.

Exemple:

```
<fwk:deleterows source="deleteRowsButton" target="taulaId" listName="accounts" selectionProperty="selected"/>
```

En l'exemple anterior s'assigna al botó amb id 'deleteRowsButton' la funció d'eliminar tots els registres seleccionats de la taula.

2.5.6 AddRowTag

El tag AddRowTag serveix per afegir nous registres a un llistat editable.

Funcionament

Aquest fa una única petició a servidor d'obtenir la plantilla de fila nova. En aquesta es podran posar en els camps el valors per defecte de cada camp.

Aquesta petició es fa amb el reqCode addRowEditList. Si l'action hereta del DispatchActionSuport en té un de genèric. Aquest construeix el bean pare buit amb la col·lecció indexada amb un element buit, i llavors fa el forward indicat per a que es construeixi la plantilla de fila nova. Aquest mètode es pot redefinit per tal que es creï un element dins la col·lecció amb els valors per defecte desitjats.

La plantilla de fila nova té unes característiques especials.

Les propietats indexades han de fer referència a l'element 0 de la col·lecció. Això és causa ja que openFrame abans de renderitzar la plana omple la col·lecció amb un bean buit. Si és fes referència a un element superior a 0 en la plantilla sortirà una excepció.

La variable {rowIndex} es reemplaça just en el pintat per la posició que ocupar en la llista indexada del bean. És molt útil per poder fer referència a component dins la mateixa fila, com és el cas de les validacions.

Per tal que els components caixes de text es pintin és necessita que estiguin dins un formulari. I aquest tingui dins un table amb un tr, que és el que es pintarà com a plantilla de fila nova.

Per tal que les validacions obligatòries de submit funcionin, es necessita que aquest formulari tingui el mateix identificador intern que el que conté el llistat editable. Per fer això s'ha d'afegir al formulari de la plantilla nova el següent:

```
generateId="<%=request.getParameter("formIdent")%>"
```

Un cop el navegador té la plantilla de fila nova del servidor, a cada petició d'afegir fila nova el que fa és copia aquesta plantilla i li reemplaça la propietat indexada amb el valor 0 pel que li correspon en aquell moment en el llistat editable.

Propietats del tag

Les propietats d'aquest tag són les següents:

Propietat	Requerit	Valor
listName	Si	Nom de la llista en el Form Bean.
source	Si	Nom de l'element disparador de l'event.
target	Si	Id de la taula que conté el llista editable.
insertInRow	Si	Posició dins la taula en la qual volem afegir els nous registres.
startIndex	Si	Índex a partir del qual afegirem nous registres en la llista.
url	Si	Url on es troba l'action que retornarà la jsp que conté la plantilla del TR a afegir.
reqCode	No	Nom de la funció dins l'Action indicada en la propietat 'url'.
parameters	No	Paràmetres addicionals per afegir a la url.
forward	No	Nom del forward que s'executarà en la funció de la Action.

- **source**

Aquesta propietat defineix l'id de l'element al qual s'assignarà l'event 'onclick' que dispararà la funció d'afegir un nou registre.

- **target**

Id de la taula on es crearan els nous registres.

- **insertInRow**

Posició gràfica de la taula on afegirem les noves plantilles. Els possibles valors poden ser:
· first
· second

· third
· last
· penultimate
· qualsevol nombre enter, tenint en compte que 0 és la primera posició i que les nombres negatius són el número de fila començant pel final de la taula. Així, si la propietat val -1, s'afegiran els nous registres a la penúltima posició de la taula, si val -2, l'avantpenúltima, etc..

- **startIndex**

Índex de la propietat indexada a partir del qual s'aniran renombrant les noves files. Lo més habitual és que l'startIndex sigui el nombre d'elements de lllista. Com que normalment no es sap s'ha d'emprar un variable que faci de comptador dins el jsp.

url

Els nous registres(o TRs) que s'han d'afegir a la taula es crearan a partir una plantilla. Aquesta plantilla serà una jsp on les propietats indexades sempre tindran index 0. El tag ja s'encarregarà de substituir el 0 per l'índex apropiat. La propietat 'url' indica la Action on es troba aquesta jsp, que contindrà una taula amb el TR que vulguem definir i que haurà de tenir la mateixa estructura que tots els altres registres (o TRs) de la taula.
Exemple:

```
  <table>
   <tr onmouseover="this.className='selected';"
onmouseout="this.className='zebra0';">
          <td>
                 <input type="checkbox" name="accounts[0].selected"/>
          </td>
          <td>
                 <input type="text" name="accounts[0].lastname"/>
          </td>
   </tr>
  </table>
```

En l'exemple anterior definim una taula que conté la plantilla dels nous registres. Si en el tag hem informat la propietat 'startIndex' amb el valor 2, en afegir un nou registre el tag substituirà 'accounts[0]' per 'accounts[2]'; si n'afegim un altre, ho farà per 'accounts[3]', etc..

- **reqCode**

Nom de la funció dins la Action definida en 'url' que gestionarà l'enviament de la plantilla jsp.

- **forward**

Nom del forward que envia la plantilla jsp. Per defecte serà 'success'.

Exemple:

```
<fwk:addrow url="/openFrame-samples-jPetstore/AppJava/accounts.do"
                        target="taulaId"
                        source="triggerButton"
                        listName="accounts"
                        startIndex="${count}"
                        insertInRow="penultimate"
                        reqCode="addRowEditList"/>
```

En l'exemple anterior s'assigna al botó amb id 'triggerButton' la funció d'afegir els nous registres de la taula.

Exemple complet de llistat editable amb valuelist:

```
<%@ include file="/WEB-INF/jsp/includes/fwkTagLibs.jsp" %>
 <br>
<script src="/openFrame-samples-jPetstore/scripts/ajax/ajaxtags/openFrame-
ajaxtags-editList.js" type="text/javascript"></script>
<script src="/openFrame-samples-jPetstore/scripts/ajax/json/json.js"
type="text/javascript"></script>

<c:set var="count" value="0"/>

<fwk:vlhroot value="list" url="accounts.do?" includeParameters="reqCode"
configName="vlConfig">
   <fwk:form styleId="actionForm" action="accounts.do" reqCode="search">
   <fwk:setList listProperty="accounts" scopeKey="list" scope="page"/>
   <fwk:defineProperty listProperty="accounts"
itemsClass="net.opentrends.openframe.samples.jpetstore.model.Account"/>

   <table width="450" align="center" border="0">
          <!-- pagination section -->
          <tr>
                 <td align="left" nowrap="true"><c:out
                        value="${list.valueListInfo.totalNumberOfEntries}" />
<bean:message key="jsp.accounts.accountList.total"/> - <bean:message
key="jsp.accounts.accountList.page"/>
                 (<c:out value="${list.valueListInfo.pagingPage}" />
<bean:message key="jsp.accounts.accountList.of"/> <c:out
                        value="${list.valueListInfo.totalNumberOfPages}" />)</td>
```

```
                <td ><table border="0" align="right"><tr><td><fwk:vlhpaging
/></td></table></td>
          </tr>
          <!-- end of Pagination -->
          <tr>
                <td colspan="2">
                <table width="100%" class="classicLook" cellspacing="0"
                      cellpadding="0" id="taulaId">
                      <tbody>
                      <%
                            String displayProvider =
(request.getAttribute("displayProvider")!=null)?request.getAttribute("displayProvi
der").toString():"";
                      %>
                      <fwk:vlhrow bean="account"
display="<%=displayProvider%>">

                            <vlh:attribute
name="onmouseover">this.className='selected';</vlh:attribute>
                            <vlh:attribute
name="onmouseout">this.className='zebra0';</vlh:attribute>
                            <fwk:vlhcolumn
titleKey="jsp.accounts.accountList.lastname" property="lastname">
                                <fwk:checkbox
property="accounts[${count}].selected" styleId="accounts[${count}].selected"
layout="false"/>
                            </fwk:vlhcolumn>
                            <fwk:vlhcolumn
titleKey="jsp.accounts.accountList.id" property="id">
                                  <fwk:vlhaction url="editAccount.do?">
                                        <fwk:vlhaddParam name="reqCode"
value="edit" temp="false" />
                                        <fwk:vlhaddParam name="id"
property="id" temp="false" />
                                        <c:out
value="${account.id}"></c:out>
                                  </fwk:vlhaction>
                            </fwk:vlhcolumn>
                            <fwk:vlhcolumn
titleKey="jsp.accounts.accountList.firstname" property="firstname"/>
                            <fwk:vlhcolumn
titleKey="jsp.accounts.accountList.lastname" property="lastname">
                                <fwk:text
property="accounts[${count}].lastname" styleId="accounts[${count}].lastname"
layout="false"/>
                            </fwk:vlhcolumn>
                            <fwk:vlhcolumn
titleKey="forms.accountForm.field.preferredCategory"
property="preferredCategory.id">
                                  <fwk:select styleId="preferredCategory.id"
property="preferredCategory.id"/>
                            </fwk:vlhcolumn>
                            <c:set var="count" value="${count + 1}"/>
                      </fwk:vlhrow>
                      <tr>
                            <td colspan="4" style="text-align:right">
                                  <table align="right"><tr><td
style="border: 0px"><bean:message key="jsp.common.export"/> </td>
                                  <td style="border: 0px"><html:link
page="/accounts.do?reqCode=searchExportPDF" target="_blank"><img
src="<%=request.getContextPath()%>/images/icono_acrobat.gif"
border="0"/></html:link></td>
                                  <td style="border: 0px"><html:link
page="/accounts.do?reqCode=searchExportExcel" target="_blank"><img
src="<%=request.getContextPath()%>/images/icono_excel.gif"
border="0"/></html:link></td>
                            </td>
```

```
                        </tr>
                    </tbody>
                </table>
                </td>
            </tr>
    </table>

    <input type="button" name="triggerButton" id="triggerButton" value="Add Row"/>
    <input type="button" name="selectionButton" id="selectionButton" value="Select
All"/>
    <input type="button" name="unselectionButton" id="unselectionButton"
value="Unselect All"/>
    <input type="button" name="deleteRowsButton" id="deleteRowsButton"
value="Delete selecteds"/>

    <br>

     <fwk:selection source="selectionButton" target="taulaId" listName="accounts"
selectionProperty="selected" action="selectAll"/>
     <fwk:selection source="unselectionButton" target="taulaId" listName="accounts"
selectionProperty="selected" action="unselectAll"/>

     <fwk:deleterows source="deleteRowsButton" target="taulaId" listName="accounts"
selectionProperty="selected"/>

    <fwk:addrow url="/openFrame-samples-jPetstore/AppJava/accounts.do"
                        target="taulaId"
                        source="triggerButton"
                        listName="accounts"
                        startIndex="${count}"
                        insertInRow="penultimate"
                        reqCode="addRowEditList"/>

    <fwk:submit value="submit" styleId="submit" reqCode="save"/>

    </fwk:form>
</fwk:vlhroot>
```

Identificador	Nombre	Apellidos	Email	Ciudad
AA	Eulus	Magnus	q@w.es	DreamLand
j2ee	Carlinhos	Brown	yourname@yourdomain.com	ReusLand
Pere	Pere	Vila	a@es.es	Barcelona
Ramon	Ramon	Flavià	e@e.es	Tarragona

Add row | Delete row | Select All | Unselect All

2.6 Servei de validacions en el jsp

El dividirem en dos gran àrees. La primera serà el motor de regles, i la segona el servei de presentació de missatges d'error.

2.6.1 Motor de regles

El motor de regles és el conjunt de mètodes que permeten validar si les dades són correctes. Aquestes s'han de poder executar tant en client com en servidor.

2.6.1.1 Definició de regles

Per definir les regles de validació és fa servir el framework commons-validator 1.3.0 d'Apache. En el framework es poden veure definides en el fitxer validations-rules.xml.

```
<validator name="maxlength"
            classname="org.springmodules.validation.commons.FieldChecks"
               method="validateMaxLength"
         methodParams="java.lang.Object,
                       org.apache.commons.validator.ValidatorAction,
                       org.apache.commons.validator.Field,
                       org.springframework.validation.Errors"

              depends=""
                  msg="errors.maxlength">

         <javascript><![CDATA[
            function validateMaxLength(form) {
                var isValid = true;
                var focusField = null;
                var i = 0;
                var fields = new Array();
                oMaxLength = new maxlength();
                for (x in oMaxLength) {
                    var field = form[oMaxLength[x][0]];

                    // openFrame modification. Added if. In some cases, a 'extend'
                  // field appears that is not a field.
                  if (field) {


                        if (field.type == 'text' ||
                            field.type == 'textarea') {

                            var iMax = parseInt(oMaxLength[x][2]("maxlength"));
                            if (field.value.length > iMax) {
                                if (i == 0) {
                                    focusField = field;
                                }
                                fields[i++] = oMaxLength[x][1];
                                isValid = false;
                            }
                        }
                  }
                }
                if (fields.length > 0) {
                   focusField.focus();
                   alert(fields.join('\n'));
                }
```

```
                return isValid;
            }]]>
        </javascript>

    </validator>
```

Les regles que tenen definida la funció javascript permeten l'execució en client, però sempre han de tenir la implementació en java per garantir l'execució en servidor.

Les regles poden dependre d'un o més camps. Aquelles que necessiten més d'un camps per validar en direm que són regles amb camps dependents. Per tant se li ha d'indicar d'alguna manera aquesta necessitat. Amb el commons-validator és fa mitjançant passant-li els valors dels camps com argument, en canvi amb el nou tag de validacions s'indica amb un atribut a mida.

2.6.1.2 Events de validació

Els events que llencen el sistema de validació són els següents:

- onSubmit del formulari:

Quan es fa un submit del formulari és llença una petició de validació via AJAX cap a servidor o s' executa regles de javascipt si es validació client . Si aquesta és correcta es fa el submit.

Per qüestions de seguretat, quan es fa el submit en servidor es torna a validar les dades que són les que entraran a l'action. Per exemple si es desactivés el javascript les validacions AJAX no funcionarien, però al baixar les dades és validaran per les validacions obligatòries de submit.

Aquestes validacions de submit obligatòries són configurables i pots establir quin mètodes han de tenir validacions obligatòries.

```
<bean name="/accountList" parent="accountListBaseDefinition">
   <property name="webValidationSubmit" ref="webValidationSubmit"/>
   <property name="metodosList">
          <list>
             <value>save</value>
          </list>
   </property>
</bean>
```

En l''action accountList s'ha definit que hi ha validacions obligatòries de submit pel mètode save de l'action.

Les regles a executar per un formulari és defineixen en el validation.xml

```
<form name="accountList">
    <field property="firstname" depends="required">
         <arg0 key="forms.accountForm.field.firstname"/>
    </field>
    <field property="lastname" depends="required">
             <arg0 key="forms.accountForm.field.firstname"/>
    </field>

</form>
```

El conjunt de regles anomenades accountList obliguen ha tenir el camp firstname i lastname informats.

Altrament aquesta seria la manera antiga. Amb la nova versió és permet definir les validacions de submit d'un camp en el mateix tag del jsp. Aquesta manera és més flexible i es poden veure fàcilment les validacions d'una pàgina.

Un exemple és :

```
<fwk:text styleId="firstname" styleClass="fieldtext"
      property="firstname" validations="ONSUBMIT(required)" />

<fwk:text styleId="lastname" styleClass="fieldtext"
      property="lastname"  validations="ONSUBMIT(required)"/>
```

Aquest exemple és sinònim de l'anterior.

A l'atribut validations es posa ONSUBMIT(< regla de validació >, < regla de validació > ..) on les regles de validacions han d'estar definides en el validation-rules.xml.

- onChange d'un camp de text:

Aquest event permet validar el contingut d'un camp un cop ha canviat. Són el que s'anomenen les validacions de camp.

Aquestes validacions només permeten execució de regles en servidor, mitjançant petició AJAX.
Al produir-se l'event onChange del camp, es llença petició a servidor indicant quin camp s'ha de validar i les regles de validació a aplicar.
En el cas que tingués camps dependents, s'enviarà el valor d'aquests camps.

La manera d'expressar aquestes validacions és:

```
<fwk:text styleId="lastname" styleClass="fieldtext"
      property="lastname" validationFieldMessageMode="ICON,TEXTERROR"
      validations="ONCHANGE(required) "
      errorClass="errorNew" errorKey="forms.accountForm.field.lastname"
      iconStyleId="lastnameIconError"
      textErrorStyleId="lastnameTextError" />
```

Escrivint ONCHANGE(..<regla de validació>, <regla de validació>) en l'atribut validations és defineix les validacions camp a camp. Aquestes regles de validació han d'estar definides en el fitxer validation-rules.xml.

Els altres atributs serveixen per mostrar l'avís d'error que seguidament s'explicarà en el següent apartat.

2.6.2 Presentació de missatges

Principalment són quatre tags, dos per validacions camp a camp i els altres dos per validacions de formulari.

2.6.2.1 Form i FormValidator

El FormValidator és un decorador que permet associar a un formulari validacions. Els atributs dels dos són bastant semblants per tant s'explicaran junts.

fwk:Form: Només s'exposen atributs de validació.

Atributs	Requerit	Descripció
validationFormMessageMode	No	Pot ser PANEL, FIELDS, WINDOW
validationMessageFunction	No	Funció de presentació.
`indicator`	No	Div que es mostra quan es fa petició AJAX

Atributs	Requerit	Descripció
mode	No	CLIENT o SERVIDOR
generateId	No	Id del formulari intern.
errorPanelStyleId	No	Id del panell d'errors.

fwk:FormValidator:

Atributs	Requerit	Descripció
source	Si	Formulari a associar-li validacions.
validationFormMessageMode	No	Pot ser PANEL, FIELDS, WINDOW
validationMessageFunction	No	Funció de presentació.
`indicator`	No	Div que es mostra quan es fa petició AJAX
mode	No	CLIENT o SERVIDOR
errorPanelStyleId	No	Id del panell d'errors.

- **validationFormMessageMode**

Indica les maneres de mostrar els errors de formulari. Hi ha 3 maneres possibles.
PANEL: Panell incrustat en la pàgina.
WINDOW: Mostra un alert amb els missatge d'error
FIELDS: Mostra els errors com si fos un validació camp a camp.

Es poden combinar els tres elements, encara que PANEL i WINDOW poden semblar contradictori, com per exemple PANEL,FIELDS;WINDOW o PANEL,FIELDS.

- **validationMessageFuncti**on

Funció javascript que s'encarrega de fer la presentació del errors. Aquest és pels usuaris que volguessin fer un presentació d'errors a mida.

- **indicator**

Div que es mostra mentre s'està fent petició a servidor via AJAX.

- **mode**

Aquest atribut és necessari per quan no estant configurades les validacions del commons validator i només hi ha validacions dels tags.

El seu valor pot ser CLIENT o SERVER, i indica on s'executaran les validacions.
En el cas que estiguessin configurades les validacions del commons-validator, agafarà el mode de l'atribut validationType.

Si l'atribut mode i el validationType no fossin iguals saltarà una excepció avisant del conflicte.

- **errorPanelStyleId**

Identificador del panell incrustat. Si no s'identifica fa servir el per defecte.

2.6.2.2 Text i FieldValidator

fwk:text: Només es mostren els atributs referents a validacions.

Atributs	Requerit	Descripció
styleId	Sí	Id d'enllaç amb la configuració XML.
validations	Si	validacions pel camp,
validationFieldMessageMode	Si	ICON,TOOLTIP,CHANGESTYLE,TEXTERROR,
sourceErrorTooltip	No	Pot ser ICON, TEXT
iconStyleId	No	Identificador de l' icona
textErrorStyleId	No	Div amb el text d'error.
errorClass	No	Estil de la classe quan està en error.
validationMessageFunction	No	Funció de presentació
dependentFields	No	Camps dependents
errorKey	Si	Nom del camp internacionalitzat.
Indicator	No	Div a mostrar quan es fa petició AJAX.

f**wk:fieldValidator**: permet que a qualsevol camp de text se li puguin associar validacions

Atributs	Requerit	Descripció
styleId	Sí	Usar el valor 'configurationTag'
Source	Sí	Component a associar les validacions.

Atributs	Requerit	Descripció
validations	Si	validacions pel camp,
validationFieldMessageMode	Si	ICON,TOOLTIP,CHANGESTYLE,TEXTERROR,
sourceErrorTooltip	No	Pot ser ICON, TEXT
iconStyleId	No	Identificador de l' icona
textErrorStyleId	No	Div amb el text d'error.
errorClass	No	Estil de la classe quan està en error.
validationMessageFunction	No	Funció de presentació
dependentFields	No	Camps dependents
errorKey	Si	Nom del camp internacionalitzat.
indicator	No	Div a mostrar quan es fa petició AJAX.

- **validations**

Cadena que indica les validacions d'un camp. La cadena permesa és:
ONCHANGE(r1,r2,...), ONSUBMIT(r3,r4,..)
Les regles de validació r1 i r2 s'executaran amb l'event onChange, i les regles r3 i r4 s'executaran en l'onSubmit del formulari.

r1,r2,r3,r4 han d'estar definides en el validation-rules.xml.

- **validationFieldMessageMode**

Els valors permesos és un combinació separada per comes dels següents valors.

TOOLTIP: És mostra un tooltip amb el missatge d'error en el camp o en l' icona.
ICON: Icona d'avís que el camp té un error. Cal informar iconStyleId.
TEXTERROR: Mostra el missatge en un div. Cal informar textErrorStyleId.
CHANGESTYLE: Canvi l'aspecte del component. Cal informar errorClass.

Per exemple, si fos TOOLTIP, ICON, mostra un tooltip i una icona.

- **sourceErrorTooltip**

Indica on és vol mostrar el tooltip. Els valors possibles són dos:
ICON: Mostrar tooltip en l' icona
TEXT: Mostrar tooltip en el camp.

- **iconStyleId**

Identificador del div que conté l' icona. El tag iconError és un tag fet a mida per mostrar icones.

- **textErrorStyleId**

Identificador del div que mostra el missatge d'error incrustat. El tag textError ajuda en aquesta tasca.

- **errorClass**

Nom de classe CSS que s'aplica en cas d'error en el camp. Normalment posa el marc en vermell del camp i canvia el color de fons.

- **validationMessageFunction**

Funció javascript que permet la presentació del missatge d'error a mida.

- **dependentFields**

Llista de camps dependents. Per exemple pot ser

<fwk:text property="age" dependentFields="currentDate,birdhDate" />

Els camps són les properties dels camps de text.

- **errorKey**

Argument que es s'envia per construir el missatge d'error.
Generalment els missatges d'error es construeixen amb una variable
Per exemple:
El camp {0} es obligatori.

Suposem errorKey = ' ciutat'
Llavors el missatge d'error serà El camp ciutat es obligatori.

Si hi ha les validacions activades aquest camp es obligatori.

- **indicator**

Identificador del div que es mostra quan hi ha petició a servidor.

2.6.2.3 IconError i TextError

Aquests dos tags són d'ajuda per tal de facilitar la mostra d'errors del sistema de validació. Mostren molt clarament la funcionalitat dels divs. Són molt aconsellables ja que ajuden es veu clarament la funcionalitat de cada tag.

L'IconError s'encarrega d'encapsular la imatge d'avís d'error.

Atributs	Requerit	Descripció
styleId	Sí	Id del div
styleClass	No	CSS del div
style	No	Style del div
services	No	Permet injectar serveis del contenidor d' spring.

Exemple

El TextError és un div que permet afegir-hi el missatge d'error.

Atributs	Requerit	Descripció
styleId	Sí	Id del div
styleClass	No	CSS del div
style	No	Style del div
services	No	Permet injectar serveis del contenidor d' spring.

Exemples:

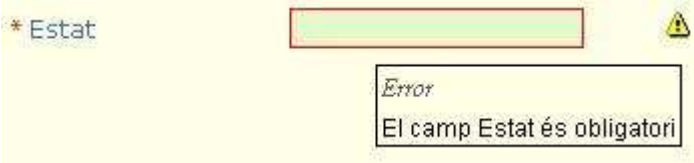
·ICON & TEXTERROR

```
                                        <td><fwk:text styleId="lastname"
styleClass="fieldtext"
                                                    property="lastname"
validationFieldMessageMode="ICON,TEXTERROR"
   validations="ONCHANGE(required),ONSUBMIT(required)"
```

```
                                                    sourceErrorTooltip="TEXT"
dependentFields="firstname"
                                                    errorClass="errorNew"
errorKey="forms.accountForm.field.lastname"
                                                    iconStyleId="lastnameIconError"

   textErrorStyleId="lastnameTextError" /></td>
                                                <td><fwk:iconError
styleId="lastnameIconError"
                                                    style="display:none"
styleClass="iconError">
                                                    <img src="<c:url
value="/images/iconWarning.gif"/>"
                                                            id="imagen" class="icon" />
                                                </fwk:iconError></td>
                                                <td><fwk:textError
styleId="lastnameTextError"
                                                    styleClass="errorText" /></td>
```

- ICON, TOOLTIP & CHANGESTYLE

```
                                                <td><fwk:text styleId="state"
property="state"
                                                    styleClass="fieldtext"

   validationFieldMessageMode="ICON,TOOLTIP,CHANGESTYLE"

   validations="ONCHANGE(required),ONSUBMIT(required)"
                                                    sourceErrorTooltip="TEXT"
dependentFields="firstname"
                                                    errorClass="errorNew"
errorKey="forms.accountForm.field.state"
                                                    iconStyleId="stateIconError"
textErrorStyleId="stateTextError" />
                                                </td>
                                                <td><fwk:iconError
styleId="stateIconError" style="display:none"
                                                    styleClass="iconError">
                                                    <img src="<c:url
value="/images/iconWarning.gif"/>"
                                                            id="imagen" class="icon" />
                                                </fwk:iconError></td>
                                                <td><fwk:textError
styleId="stateTextError" styleClass="errorText" />
                                                </td>
```



2.6.3 Validacions en llistats editables

Les validacions en llistats editables és el suport a propietats indexades. És a dir poder validar camps amb els noms de propietats a[1].id o a.b[2].name,etc. Els [] denoten propietats indexades i són pròpies de llistats editables.

Seguidament és mostra un exemple de validació en propietats indexades.

```
<fwk:text styleId="city" property="accounts[0].city" size="20"
                    validationFieldMessageMode="ICON,TOOLTIP,CHANGESTYLE"
validations="ONCHANGE(required),ONSUBMIT(required)" sourceErrorTooltip="TEXT"
iconStyleId="iconErrorCity${count}" errorKey="forms.accountForm.field.city"
errorClass="errorNew"/>
```

Seguidament s'explicarà les peculiaritats segons el mètode de validació.

-onChange:

Per aquest sistema no hi ha absolutament cap problema ja que el camp és un etiqueta per aquest sistema de validació i li és igual com s'escrigui.

-onSubmit per commons validator:

No s'aconsella pel motius seguidament explicats.

Les propietats indexades s'han d'especificar amb l'atribut indexedListProperty i s'ha d'especificar el nom de la propietat que és una llista.

```
<form name="accountList">
    <field property="firstname" indexedListProperty="accounts" depends="required">
            <arg0 key="forms.accountForm.field.firstname"/>
    </field>
    <field property="lastname" indexedListProperty="accounts" depends="required">
            <arg0 key="forms.accountForm.field.firstname"/>
     </field>

</form>
```

El problema d'aquest sistema és que el sistema de presentació d'errors rep el camp erroni però no s'indica la fila errònia i per tant no es pot saber quin camp està malament.

Per exemple en el cas anterior, es rebria com error accounts[].city.

-onSubmit per tags.

Si es defineix l'event onSubmit en els tags el problema sorgeix en el mecanisme d'afegir files noves.

Aquest mecanisme dispara un única petició a servidor en busca de la plantilla nova d' afegir. Un cop la té el navegador reemplaça el nomCollecció[0], pel número nomColecció[ultimIndex].
Aquest últim pas es va repetint tants cops com addRows es facin.

El problema d'aquest mecanisme es que les validacions de submit s'associen a un identificador intern de formulari i es queden a sessió, i per tant per les noves files no hi ha associades validacions de submit.
Llavors s'ha de fer dos passes.

1. La primera es indicar-li al servidor que tenim una propietat indexada i que totes les validacions de submit de la posició X, generalment la 0, es propaguin a totes les files de la col·lecció.

Això és fa amb el tag validateCollection.

Atributs	Requerit	Descripció
property	Sí	Nom de la propietat que és col·lecció
indexReference	No	Numero de fila referència per a propagar les validacions de submit. Si no s'informa s'assumeix 0.

Exemple:

```
<fwk:validateCollection property="accounts" indexReference="0"/>
```

2. Si en la plantilla de files noves (addRow.jsp) és tenen tags del fwk, segurament caldrà un formulari. En el cas de definir validacions de submit en aquesta plantilla, s'haurà d'informar en el formulari l'atribut generateId.

```
<fwk:form styleId="actionForm11"
   action="accountList.do"       reqCode="save"
   generateId="<%=request.getParameter("formIdent")%>"
   services="logService:loggingService" >
```

2.6. Utilització del Servei

2.6.1. Ús dels tags a les pàgines JSP

L'ús de cadascun dels tags implica 2 passos:

1) Definició de la referència a la llibreria de tags

Des de la versió JSP 1.2 no és necessari configurar al fitxer web.xml la llibreria i podem fer referències directes mitjançant url al jar, tal i com es mostra a continuació:

```
<%@ taglib prefix="fwk"  uri="http://www.opentrends.net/openframe/open-layout" %>
```

2) Definició del tag a la pàgina JSP

Una vegada definida la referència a la llibreria i el seu prefix 'fwk' podem incorporar qualsevol tag mitjançant '<fwk:nomTag>'.

Per tots els tags d'entrada de dades de formulari es definiran els atributs:

Propietat	Requerit	Valor
styleId	Sí	Identificador del camp. Aquest identificador és el que correspon a la propietat 'styleId' en el fitxer de configuració del tag
property	Sí	Nom del atribut del bean associat al formulari que volem mostrar. (l'associació del bean es fa mitjançant la propietat 'pojoClass' del fitxer de configuració de l'acció)

Exemple:

```
<fwk:text styleId="firstname" property="firstname"/>
```

En la següent taula es dona un resum dels tags permesos i la correspondència amb la seva classe que s'ha de definir en el fitxer de configuració:

Tag	Classe
fwk:actionImage	net.opentrends.openframe.services.web.struts.taglib.forms.fields.ActionImageTag
fwk:checkbox	net.opentrends.openframe.services.web.struts.taglib.forms.fields.CheckboxFieldTag
fwk:configuration	net.opentrends.openframe.services.web.struts.taglib.configuration.ConfigurationTag
fwk:file	net.opentrends.openframe.services.web.struts.taglib.forms.fields.FileFieldTag
fwk:form	net.opentrends.openframe.services.web.struts.taglib.forms.fields.FormTag
fwk:options	net.opentrends.openframe.services.web.struts.taglib.forms.fields.OptionsFieldTag
fwk:password	net.opentrends.openframe.services.web.struts.taglib.forms.fields.PasswordFieldTag
fwk:radio	net.opentrends.openframe.services.web.struts.taglib.forms.fields.RadioFieldTag
fwk:select	net.opentrends.openframe.services.web.struts.taglib.forms.fields.SelectFieldTag
fwk:submit	net.opentrends.openframe.services.web.struts.taglib.forms.fields.SubmitTag
fwk:text	net.opentrends.openframe.services.web.struts.taglib.forms.fields.TextFieldTag
fwk:textarea	net.opentrends.openframe.services.web.struts.taglib.forms.fields.TextAreaFieldTag

2.6.2. Comentaris Específics a alguns Tags

En l'ús del tag 'select' es recomana l'ús de JSTL per a la generació de les opcions incloses.

Exemple:

```
<fkw:select styleId="..." property="...">
```

```
    <c:forEach items="${nombreLista}" var="item">
      <fwk:option value="${item}">${item.descripcion}</fwk:option>
    </c:forEach>
  </select>
```

2.6.3. Especificació del Mode del Formulari

En la configuració dels components d'entrada de formulari s'ha explicat com podem mostrar-los de forma diferents segons el mode del formulari. Aquest mode de formulari pot canviar-se mitjançant la crida:

FormUtils.setFormDisplayMode(request, actionForm,mode);

On mode pot ser un dels següents valors:

- FormUtils.INSPECT_MODE. Mode de consulta
- FormUtils.EDIT_MODE. Mode d'edició
- FormUtils.CREATE_MODE. Mode de creació

NOTA:
openFrame, dins la seva classe 'ActionExtendedSupport' (classe pare de les nostres action) efectua aquesta tasca per a que no ens haguem de preocupar. Per a això utilitza un *resolver* que segons el reqCode que arribi per paràmetre decidirà el model del formulari. A continuació es mostra una taula amb els modes usats segons el reqCode:

```
public static final String EDIT_METHOD_PREFIX = "edit";
public static final String SEARCH_METHOD_PREFIX = "search";
public static final String NEW_METHOD_PREFIX = "create";
public static final String DELETE_METHOD_PREFIX = "delete";
public static final String SAVE_METHOD_PREFIX = "save";
public static final String VIEW_METHOD_PREFIX = "view";
```

reqCode	Mode utilitzat
edit	FormUtils.EDIT_MODE
search	FormUtils.INSPECT_MODE
create	FormUtils.CREATE_MODE
view	FormUtils.INSPECT_MODE
En altres casos	FormUtils.EDIT_MODE

2.7. Eines de Suport

2.7.1. Debug de les Pàgines

En l'ús de les nostres pàgines és convenient sempre conèixer si estan ben construïdes. Tal i com es comenta al document 'Entorn de Desenvolupament' mitjançant l'extensió 'Web Developer' i 'Javascript Debugger' podem esbrinar i detectar problemàtiques de les nostres pàgines.

2.8. Integració amb Altres Serveis

2.9. Preguntes Freqüents

3. Exemples

4. Annexos

4.6. JSTL

Des de fa un temps existeix **JSTL** (JSP Standard Tag Library). Aquesta llibreria de tags defineix algunes de les tasques bàsiques que necessitem a les nostres pàgines JSP (condicionals, presentació de valors, internacionalització, ...). Si bé no ofereix totes les necessitats que requerim d'una aplicació Web sí ofereix la base del que prèviament es feia amb codi Java o els tags de Struts.

Es recomana l'ús dels tags JSTL de 'core' i 'fmt' que es poden referenciar dins les nostres pàgines com:

```
<%@ taglib prefix="c"   uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt" %>
```

Podeu veure un resum de JSTL a la url 'http://www.1x4x9.info/files/jstl/html/online-chunked'.

4.7. Altres Tags

Tot i l'oferta de tags de openFrame, existeixen altres necessitats, com per exemple l'ús de pestanyes.

En aquest apartat s'ofereixen una llista de components disponibles a la comunitat que poden resoldre aquestes necessitats:

- Tags de Crumbs

Per a mostrar el típic tag d'enllaços a on ens trobem dins la pàgina podem usar el tag de Crumbs de 'Struts Layout'.

[Categorías > **Editar categoría**]

Consultar 'http://struts.application-servers.com/doc/tags/crumbs.html' per conèixer la seva utilització.

- Tags de Pestanyes

Per a utilitzar pestanyes es recomana la utilització del tag de 'Ditchnet'[3]

4.8. Aspectes d'Usabilitat

Aquesta secció pretén donar una visió general del quins aspectes s'han tingut en consideració en els components openFrame per a complir amb normes bàsiques d'usabilitat i accessibilitat.

Algunes de les guies proporcionades poden resultar òbvies però considerem la necessitat de fer-les esmentar per a tenir en consideració. En qualsevol cas, moltes de les guies explicades es troben implementades per openFrame en els seus tags, pel que no s'hauran de tenir en consideració per al desenvolupament de les aplicacions Web. Si més no, també és el nostre propósit mostrar per què s'han pres determinades decisions basades en les guies aquí establertes.

A continuació es detallen les guies utilitzades:

1) La descripció del component d'entrada hauria de precedir al component (excepte els radio i els checkbox)
2) Usar l'etiqueta 'label' i l'atribut 'for' per a fer referència al component que descriu.
   (1) i (2) són realitzats pels tags openFrame.

3) Els camps de tipus text haurien de ser seleccionats quan reben el focus

   openFrame permet especificar la propietat 'selectOnFocus' per a lligar aquest comportament als nostres components. Per defecte es troba a 'true'.

4) Els camps haurien de definir la seva longitud màxima mitjançant les propietats 'maxLength', 'width', 'size' i 'cols'

   Aquestes propietats han de ser definides pels desenvolupadors.

5) En els camps requerits haurem d'utilitzar alguna representació per indicar que són obligatoris. Si es fa servir una icona o símbol (*) indicar a la pàgina què significa

openFrame mostra, per defecte, un asterisc a l'esquerra de l'etiqueta del component si el seu atribut associat ha estat definit com a obligatori dins el Servei de Validació.

6) Quan anem a entrar dades a un component hauríem de mostrar alguna decoració que indiqui que estem en aquell component

   Per exemple podem usar el següent estil (basat en 'theserverside.com'):

```
form input, form textarea, form select {
```

3 http://209.61.157.8:8080/taglibs/

```
    padding-left: 4px;
    color: #666;
}
```

O bé:

```
input.input-text, textarea { border: 1px solid #859085; background: transparent;
padding: 1px 3px; }
input.hover, textarea.hover { border-color: #000; background-color: #ffe; }
input.active, textarea.active { border-color: #f00; background-color: #fff; }
input.input-button, button, select { border-width: 1px; padding: 0px; }
input.input-text, input.input-button, button, select, textarea { font: 90%
tahoma,verdana,sans-serif; }
input.input-disabled { background-color: #ccc; border-color: #999; color: #777; }
```

4.9. CSS (Cascade Style Sheets)

Tot i que no és propòsit de openFrame oferir un tutorial en detall de què són els CSS i com fer-los servir es considera bàsic el seu coneixement.

A continuació s'ofereix un resum de quins aspectes es consideren important a tenir en compte:

4.9.1. Importació de fitxers d'estils

Podem agrupar l'ús de varis fitxers d'estils mitjançant la sentència '@import'.

```
<style type="text/css" media="all">
@import "file1.css";
@import "file2.css";
</style>
```

1.3.3 4.4.2. Utilització de ids i classes

Mitjançant l'ús d'identificadors de classe (incorporant '.' al davant del nom de la classe) podem definir quin serà l'estil a aplicar a tots els tags que incorporin aquest nom de class (en els tags openFrame correspon a la propietat 'styleClass').

```
.required {font-family: Verdana; font-size: 12pt; color: red; }
```

Aquest estil s'aplicarà a tots els components que tinguin com a class='required', siguin divs, spans, ...

En cas que volem aplicar un estil a un únic component farem servir identificadors. És a dir, partint del id assignat al component (en openFrame l'atribut 'styleId' dels components), podem assignar un estil usant '#':

```
#header {width: 90%; background: white; font-size: 20px; color: purple; }
```

```
<div id="header">...</div>
```

Sempre podem definir els estils al mateix component (atribut 'style') o sobreescriure els definits al fitxer d'estil:

```
<span class="caution" style="color: green">text</span>
```

Si volem definir el mateix estil a diferents ids o classes podem separar cadascun d'ells amb espai tal i com es mostra en el següent exemple:

```
p b {color: red; }
```

Inclús podem definir que tots els components inclosos a un id o class determinat actuïn de forma determinada:

```
div#navigation a {color: white; }
```

En aquest cas, s'aplicarà l'estil a tots els '<a href...' que estiguin inclosos a un div amb id='navigation'

4.4.3. Efectes Comuns

En la següent taula es mostra un resum d'algunes de les propietats més útils a definir:

Efecte	Configuració
Text en itàlica	font-style: italic; (tornar a normal amb font-style:normal)
Text en negreta	font-weight: bold; (tornar a normal amb font-weight:normal)
Text en normal	
Capitalització	• Tot a majúscules: text-transform: uppercase; • Tot a minúscules: text-transform: lowercase;

	• Primera lletra a majúscules: text-transform: capitalize; • Normal:text-transform: normal;
Enllaç sense decoració	**a** {text-decoration: none; }
Marges	margin: 20px; margin-left: 2px; margin-top: 80px; margin-right: 45px; margin-bottom: -5px; **IMPORTANT**: Des de que va sortir HTML el body incorpora un espai addicional que podem eliminar amb: **body** {margin: 0px; padding: 0px; }
Padding	S'aplica entre els elements padding: 6px; padding-bottom: 2px; padding-left: 18px;
Color de Background als elements	background-color: green; background-color: rgb(0,108,64);
Imatge de background als elements	background-image: url(images/required.gif);

4.4.4. Pseudo Classes

Tot i que podem definir selectors o estils a la majoria d'elements, hi ha parts que no són elements en sí sinó parts d'aquests. Mitjançant pseudo classes podem accedir a aquests elements.

Com a guia de openFrame es proporciona al fitxer d'estils una pseudo classe per presentar els components d'entrada de diferent forma quan ubiquem el focus.

```
form input, form textarea, form select {
    padding-left: 4px;
    color: #666;
}
```

```
/* Warning: IE doesn't support pseudo-class :focus */
form input:focus, form textarea:focus, form select:focus {
    border-bottom: #ffdead solid 2px;
    border-right: #ffdead solid 2px;
    border-left: #c07300 solid 2px;
    border-top:  #c07300 solid 2px;
    color: #000;
}
```

NOTA: Internet Explorer no suporta pseudo classes