



---

## **Framework Corporatiu J2EE**

### **Servei de Validació**

**Versió 1.4**

Barcelona, 13 / abril / 2007



## Històric de modificacions

Data	Autor	Comentaris	Versió
13/01/2006	Atos Origin, sae openTrends	Versió inicial del document	1.0
02/03/2006	Atos Origin, SAE	Versió 1.1 d'OpenFrame	1.1
13/02/2007	Atos Origin, SAE	Versió 1.2 d'OpenFrame	1.2
16/03/2007	Atos Origin, SAE	Versió 1.3 d'OpenFrame	1.3
13/04/2007	Atos Origin, SAE	Versió 1.4 d'OpenFrame	1.4

### Llegenda de Marcadors



## Índex

<b>1.</b>	<b>INTRODUCCIÓ .....</b>	<b>4</b>
1.1.	PROPÓSIT .....	4
1.2.	CONTEXT I ESCENARIS D'ÚS .....	5
1.3.	VERSIONS I DEPENDÈNCIES .....	5
1.3.1.	<i>Versions</i> .....	5
1.3.2.	<i>Dependències Bàsiques</i> .....	5
1.4.	A QUI VA DIRIGIT .....	6
1.5.	DOCUMENTS I FONTS DE REFERÈNCIA .....	6
1.6.	GLOSSARI .....	6
<b>2.</b>	<b>DESCRIPCIÓ DETALLADA .....</b>	<b>7</b>
2.1.	ARQUITECTURA I COMPONENTS .....	7
2.1.1.	<i>Interfícies i Components Genèrics</i> .....	7
2.1.2.	<i>Components basats en Commons i Spring</i> .....	8
2.1.3.	<i>Components basats en Comunicació Web per Ajax</i> .....	8
2.2.	INSTAL·LACIÓ I CONFIGURACIÓ .....	9
2.2.1.	<i>Instal·lació</i> .....	9
2.2.2.	<i>Configuració</i> .....	9
2.3.	UTILITZACIÓ DEL SERVEI .....	19
2.4.	EINES DE SUPORT .....	20
2.5.	INTEGRACIÓ AMB ALTRES SERVEIS .....	20
2.5.1.	<i>Integració amb el Servei de Internacionalització</i> .....	20
2.5.2.	<i>Integració amb el Servei de Presentació</i> .....	21
2.6.	PREGUNTES FREQUÈNTS .....	25
<b>3.</b>	<b>EXEMPLES .....</b>	<b>26</b>
<b>4.</b>	<b>ANNEXOS .....</b>	<b>27</b>

## 1. Introducció

### 1.1. Propòsit

Quan realitzem una aplicació, existeixen sempre 2 tipus de validacions a considerar:

- Validació de dades

Inclou el tractament de l'entrada correcta dels tipus de dades, en la que no influeix l'estat de l'aplicació. Aquesta validació ha de tenir en compte, entre altres:

- 1) Si un camp és obligatori
- 2) Si un camp és numèric, cadena, etc.
- 3) Si un camp té un número mínim o màxim de caràcters
- 4) Si un camp compleix una expressió regular (format de nif, codi postal, etc.)

- Validació funcional

És aquella validació que depèn de l'estat de l'aplicació, i de les dades de negoci.

Aquesta validació queda fora de l'àmbit d'aquest document, perquè el seu tractament es realitzarà per mitjà del llançament i tractament d'excepcions de negoci (veure 'Servei d'Excepcions')

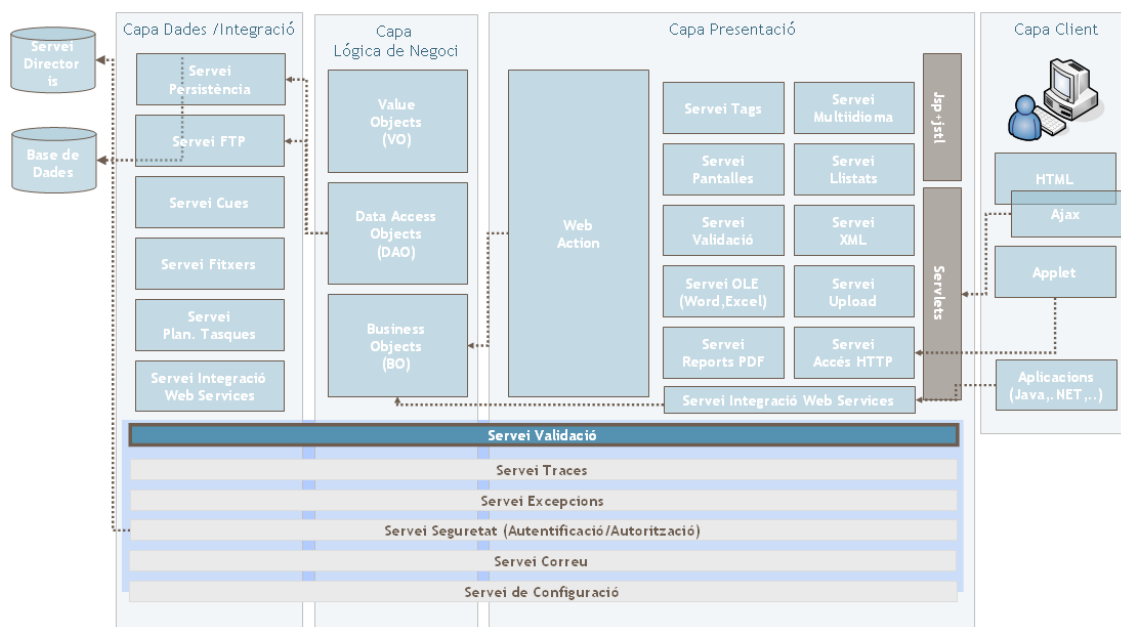
openFrame, encara que en la seva capa de presentació es basa en l'actualitat en l'ús de Struts, elimina l'ús dels ActionForm. Aquests només afegeixen complexitat innecessària i en un model vista controlador real hauriem de poder utilitzar el model de negoci des de les nostres pàgines. Per aquest motiu openFrame proporciona els mecanismes necessaris per a fer ús de classes de negoci en presentació i elimina el cost de creació de formularis Struts.

Així com en Struts la validació es realitza a nivell de presentació, en realitat el que s'està fent és anticipar la validació dels nostres objectes de negoci corresponents (un camp és requerit en presentació per què ho és en negoci, etc.).

Podem finalitzar aquest discurs posant èmfasi que el Servei de Validació és un servei de propòsit general i no únicament de presentació. En tot cas, veurem en l'apartat 'Integració amb altres Serveis' com podem realitzar la validació en presentació comunicant-nos amb aquest servei.

## 1.2. Context i Escenaris d'Ús

El Servei de Validació es troba dins dels serveis de Propòsit General de openFrame.



## 1.3. Versions i Dependències

### 1.3.1. Versions

No s'han produït canvis respecte la versió 1.3.

### 1.3.2. Dependències Bàsiques

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

Nom	Tipus	Versió	Descripció
bsf	jar	2.3.0	
commons-beanutils-core	jar	1.7.0	
commons-collections	jar	3.1	
commons-digester	jar	1.5	
commons-logging	jar	1.0.4	<a href="http://jakarta.apache.org/commons">http://jakarta.apache.org/commons</a>
commons-validator	jar	20051103	
dwr	jar	1.1-beta1	
jdom	jar	1.0	



Nom	Tipus	Versió	Descripció
junit	jar	3.8.1	
log4j	jar	1.2.12	
openFrame-core	jar	1.0	
openFrame-services-exceptions	jar	1.0	
openFrame-services-i18n	jar	1.0	
openFrame-services-logging	jar	1.0	
servlet-api	jar	2.4	
spring	jar	1.2.5	<a href="http://www.springframework.org">http://www.springframework.org</a>
springmodules-validator	jar	0.1	

## 1.4. A qui va dirigit

Aquest document va dirigit als següents perfils:

- Programador. Per conèixer l'ús del servei
- Arquitecte. Per conèixer quins són els components i la configuració del servei

## 1.5. Documents i Fonts de Referència

- [1] Manual Validator Struts [http://struts.apache.org/struts-doc-1.2.4/userGuide/dev\\_validator.html](http://struts.apache.org/struts-doc-1.2.4/userGuide/dev_validator.html)

## 1.6. Glossari

### Commons Validator

Commons Validator és un estàndar de *facto* d'API de validació

## 2. Descripció Detallada

### 2.1. Arquitectura i Components

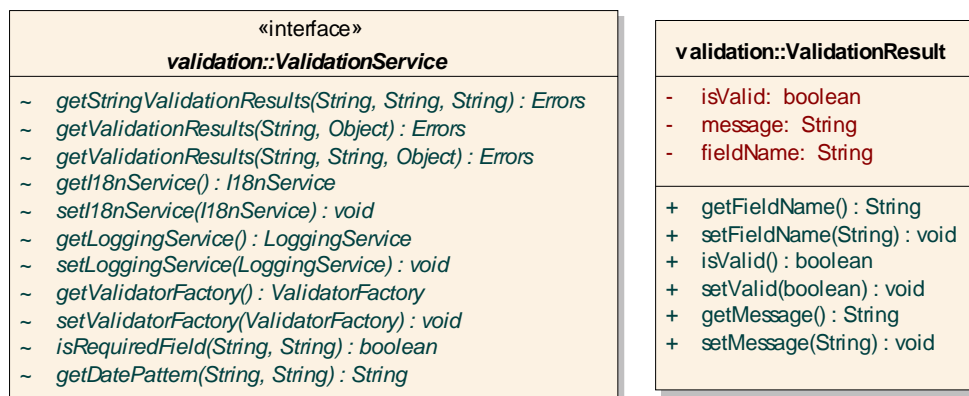
openFrame ofereix la possibilitat d'utilitzar diferents implementacions de validació mitjançant la definició d'interfícies. En l'actualitat s'ofereix una implementació basada en l'ús de Spring Commons Validator.

Els components podem classificar-los en:

- Interfícies i Components Genèrics. Interfícies del servei i components d'ús general amb independència de la implementació escollida.
- Implementació de les interfícies basada en Spring Commons Validator
- Implementació de les interfícies basada en Comunicació Web per Ajax

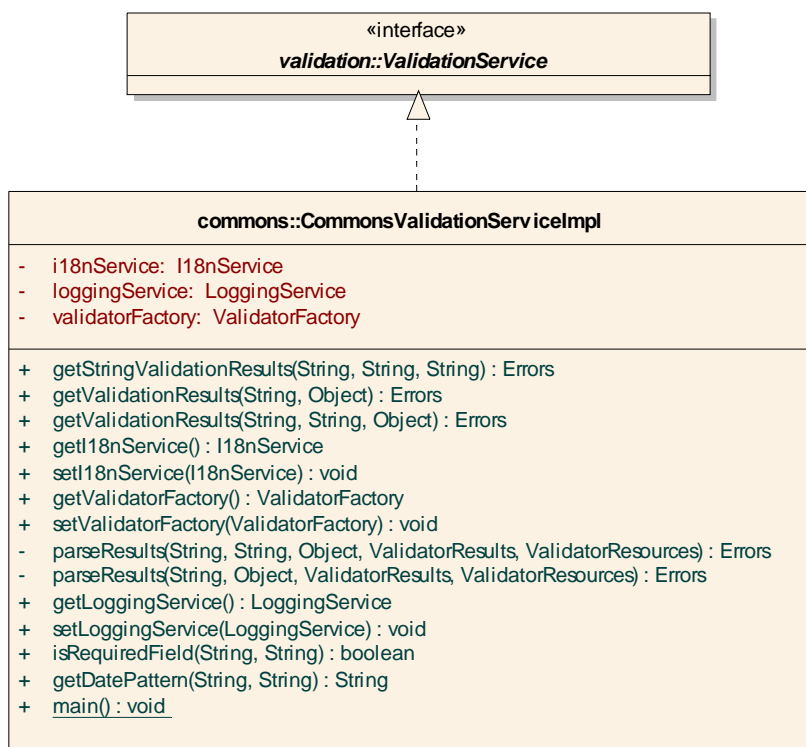
#### 2.1.1. Interfícies i Components Genèrics

El Servei de Validació defineix les següents interfícies:



Component	Package	Descripció
ValidationService	net.opentrends.openframe.services.validation	Interfície bàsica del servei
ValidationResult	net.opentrends.openframe.services.validation	Classe d'indicació del resultat de validació per un camp

### 2.1.2. Components basats en Commons i Spring

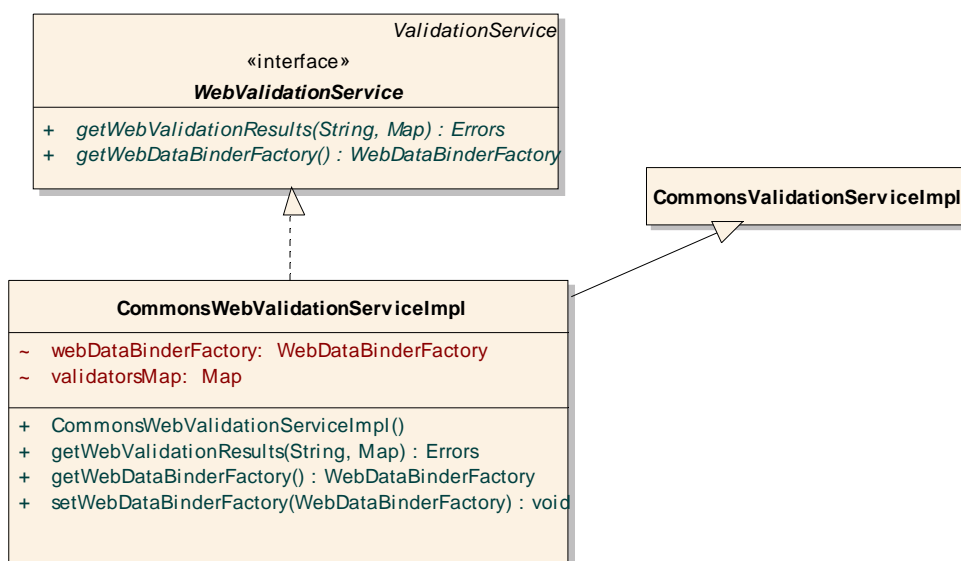


Component	Package	Descripció
CommonsValidationServiceImpl	net.opentrends.openframe.services.validation.common	Implementació basada en Commons i Spring

### 2.1.3. Components basats en Comunicació Web per Ajax

Per la validació des de clients Ajax s'ofereix una implementació específica.





Aquesta implementació es troba dins el Servei de Presentació (llibreria openFrame-services-web). En l'apartat 'Integració amb Altres Serveis' veurem la seva utilització.

## 2.2. Instal·lació i Configuració

### 2.2.1. Instal·lació

La instal·lació del servei requereix de la utilització de la llibreria 'openFrame-services-validation' i les dependències indicades a l'apartat 'Introducció-Versions i Dependències'.

### 2.2.2. Configuració

La configuració del Servei de Validació implica 3 passos:

- 1) Definir la factoria del servei
- 2) Definir quina implementació del servei s'usarà
- 3) Definir el fitxer de les regles de validació
- 4) Definir el fitxer de validacions a realitzar

Definició de la Factoria

```
<bean id="validatorFactory"
...>
```

**Fitxer de configuració:** *openFrame-services-validation.xml*

Ubicació proposada: *<PROJECT\_ROOT>/src/main/resources/spring*

La factoria permet definir la configuració de la validació. En l'actualitat s'ofereix una implementació, basada en Commons i Spring:

- `org.springframework.validation.DefaultValidatorFactory`

Per aquesta classe definirem les següents propietats:

Atributs	Requerit	Descripció
list	Sí	Llista de fitxers de validació. El contingut d'aquests fitxers s'especificarà posteriorment en el punt 3) Definir el fitxer de validacions a realitzar

En l'atribut 'list' podem especificar 2 tipus de fitxers:

1) Fitxer de regles de validació

Aquest fitxer defineix les validacions disponibles per qualsevol formulari i la classe que resol la validació. En el cas de openFrame, s'ofereix un fitxer bàsic 'validator-rules.xml' que conté les validacions més comuns.

2) Fitxer de validacions

Aquest fitxer permet definir per cada formulari o objecte quines validacions s'han de realitzar per cada camp o atribut.

Exemple:

```
<bean name="validatorFactory"
class="org.springframework.validation.DefaultValidatorFactory">
  <property name="validationConfigLocations">
    <list>
      <value>/WEB-INF/classes/validation/validator-
rules.xml</value>
      <value>/WEB-INF/classes/validation/validation.xml</value>
    </list>
  </property>
</bean>
```

#### Definició del Servei

```
<bean id="validationService"
...>
```

**Fitxer de configuració: *openFrame-services-validation.xml***

Ubicació proposada: `<PROJECT_ROOT>/src/main/resources/spring`

El bean 'validationService' és la interfície del servei que s'usarà des dels clients. En l'actualitat s'ofereix una implementació basada en Commons i Spring:

- 'net.opentrends.openframe.services.validation.common.CommonsValidationServiceImpl'. Aquesta classe requereix de la incorporació dels paràmetres següents:

Atributs	Requerit	Descripció
validatorFactory	Sí	Referència al bean definit com 'validatorFactory'
i18nService	Sí	Referència al Servei d'Internacionalització openFrame
loggingService	Sí	Referència al Servei de Logging openFrame

Exemple:

```
<bean id="validator"
class="org.springframework.commons.validator.DefaultBeanValidator">
  <property name="validatorFactory"><ref
bean="validatorFactory"/></property>
</bean>

<bean id="validationService"
class="net.opentrends.openframe.services.validation.common.CommonsValidationServiceImpl">

  <property name="validatorFactory">
    <ref bean="validatorFactory" />
  </property>

  <property name="i18nService">
    <ref bean="i18nService" />
  </property>

  <property name="loggingService">
    <ref bean="loggingService" />
  </property>

</bean>
```

## Definició de les Regles de Validació Disponibles

```
<validator name="" classname=""
...>
```

### *Fitxer sde configuració: validator-rules.xml*

Ubicació proposada: <PROJECT\_ROOT>/src/main/resources/validation

Aquest fitxer defineix les validacions que es poden realitzar.

En el fitxer 'validator-rules.xml' es defineixen els tipus de validacions que podem realitzar des d'openFrame. Aquestes regles de validació es basen en Spring i tenen la següent estructura:

```
<validator name="required"
  classname="org.springframework.validation.FieldChecks"
  method="validateRequired"
  methodParams="java.lang.Object,
    org.springframework.validation.ValidatorAction,
    org.springframework.validation.Field,
    org.springframework.validation.Errors"
  msg="errors.required">

  <javascript><![CDATA[
    function validateRequired(form) {
      var isValid = true;
      var focusField = null;
      ...

    }
  ]]>
</javascript>
</validator>
```

Cada validador defineix quina és la classe que realitza la validació (en el cas mostrat 'FieldChecks') i el mètode que es cridarà per validar el camp en el camp 'method'. Dins el tag '<javascript>' s'incorpora el codi que es generarà en el client per validar mitjançant Javascript (en el cas que la validació s'efectui des del client).

Els validadors actualment disponibles són:

Validador	Descripció
required	Validació de camps requerits
requiredif	Validació de camp requerit si es dóna una altra condició
minlength	Validació de longitud mínima de caràcters
maxlength	Validació de longitud màxima de caràcters
mask	Validació de format segons expressió regular
byte	Validació de que el camp pot ser convertit a Byte (entre $-2^7$ i $2^7-1$ )
short	Validació de que el camp pot ser convertit a Short (entre $-2^{15}$ i $2^{15}-1$ )
integer	Validació de que el camp pot ser convertit a Integer (entre $-2^{31}$ i $2^{31}-1$ )
long	Validació de que el camp pot ser convertit a Long (entre $-2^{63}$ i $2^{63}-1$ )
float	Validació de que el camp pot ser convertit a Float (entre $2^{-149}$ i $(2 \cdot 2^{-23}) \cdot 2^{127}$ )
double	Validació de que el camp pot ser convertit a Double (entre $2^{-1074}$ i $(2 \cdot 2^{-52}) \cdot 2^{1023}$ )
date	Validació de que el camp és una data correcta
intRange	Validació que un camp enter es troba dins un rang de valors



floatRange	Validació que un camp float es troba dins un rang de valors
doubleRange	Validació que un camp double es troba dins un rang de valors
email	Validació que un camp té format de correu electrònic

En el cas de voler definir un nou validador consultar la documentació de la classe 'FieldChecks'.

## Definició dels Fitxers de Validació

### ***Fitxers de configuració: validationXXX.xml***

*Ubicació proposada:* <PROJECT\_ROOT>/src/main/resources/validation

Per a millor referència de l'ús dels validadors podem consultar '[http://struts.apache.org/struts-doc-1.2.4/userGuide/dev\\_validator.html](http://struts.apache.org/struts-doc-1.2.4/userGuide/dev_validator.html)'. Tot i que la validació no es basa en 'Struts Validator' es segueix la mateixa política, pel que aquells que hi estiguin familiaritzats al seu ús podran ràpidament adaptar-se a l'ús de la validació.

En els fitxers de validació definirem quines validacions volem realitzar per als nostres objectes o formularis. Definirem 2 seccions diferenciades:

#### 1) Constants

Secció en la que es defineixen expressions regulars comunes que seran utilitzades des de diverses definicions de validació. Per exemple, la màscara corresponent a la comprovació del NIF podríem definir-la com una constant i ser referenciada en tots els camps a validar amb NIF.

```
<global>
  <constant>
    <constant-name>any</constant-name>
    <constant-value>^([0-9][0-9][0-9][0-9])$</constant-value>
  </constant>
</global>
```

#### 2) Definició dels formularis a validar

Encara que hem explicat que la validació serà a nivell d'objectes i/o formularis, Commons Validator utilitza el concepte form per a referir-se a ells. Per a cada camp especificarem quines validacions realitzar.

```
<formset>
  <form
name="net.opentrends.openframe.samples.jpjstore.persistence.Account">
    <field property="firstname" depends="required,mask">
      <arg0 key="nameForm.firstname.displayname"/>
      <msg name="mask" key="nameForm.errors.firstname.mask"/>
    </field>
  </form>
</formset>
```



```
<var><var-name>mask</var-name><var-value>^[a-zA-Z]*$</var-  
value></var>  
</field>  
  
<field property="lastname" depends="required">  
  <arg0 key="nameForm.lastname.displayname"/>  
</field>  
</form>  
</formset>
```

L'esquema d'aquest fitxer és el mostrat a dalt. Cada objecte o formulari a validar serà definit en una secció '<form>' i cada camp del formulari en una subsecció '<field>'

A l'atribut 'name' de la secció '<form>' definim el nom del validador.

Per a cada camp especificarem:

Atributs	Requerit	Descripció
property	Sí	Indicar com a valor el nom de l'atribut de la classe que volem validar
depends	Sí	Llista de validacions a realitzar separades per coma. El nom de validacions correspon al nom definit en el fitxer de regles de validació (validation-rules.xml).

Depenent del tipus de validacions definides incorporarem més propietats. A continuació detallarem per a cada tipus de validació els possibles atributs addicionals.

Abans de començar a definir les validacions és important que entenguem que existeix una diferència entre validar formularis Web i validar objectes de negoci. La diferència entre el model Web i el model de classes és que en el model Web basat en `HttpServletRequest` tots els paràmetres rebuts són de tipus 'String', mentre que en les nostres classes de negoci fem servir altres tipus. Per exemple, un clar exemple de la diferenciació el podem trobar representat en aquest diagrama:

Comparable Serializable
<b>Account</b>
<ul style="list-style-type: none"> <li>- id: java.lang.String</li> <li>- email: java.lang.String</li> <li>- firstname: java.lang.String</li> <li>- lastname: java.lang.String</li> <li>- status: java.lang.String</li> <li>- addr1: java.lang.String</li> <li>- addr2: java.lang.String</li> <li>- city: java.lang.String</li> <li>- state: java.lang.String</li> <li>- zip: java.lang.String</li> <li>- country: java.lang.String</li> <li>- phone: java.lang.String</li> <li>- birthDate: java.util.Date</li> <li>- preferredCategory: Category</li> <li>- preferredProduct: Product</li> <li>- comments: String</li> <li>- isLower18: boolean</li> <li>- password: String</li> </ul>

En aquest cas, 'birthDate' és de tipus 'Date'. Si construïm un objecte de tipus 'Account' no té cap sentit validar si 'birthDate' és una data correcta, doncs el tipus de dades ja ens lliga a que sigui una data correcta. En cas però de que fos un 'String' sí que hauríem de validar que és una data correcta. Per tant, hem de tenir en compte que no definirem cap validació de tipus data si el tipus de dada no és un String. Posteriorment, consultar 'Integració amb el Servei de Presentació' per conèixer com es podria popular un atribut de tipus 'Date' a partir d'un paràmetre Web de forma que es tornés un missatge d'error de format a l'usuari.

- **Definir camps requeris**

*Aplicable a qualsevol tipus de valor*

- 1) Definir el valor 'required' en l'atribut 'depends'.
- 2) Per defecte, el missatge que es mostrarà s'ha definit en la propietat 'msg' del tipus de validador (errors.required).

```

<validator name="required"
...
msg="errors.required"
/>

```

```
errors.required=El campo {0} es obligatorio
```

Pel fet que el missatge internacionalitzat requereix d'un paràmetre (indicat en {0}), podem passar-li el contingut a substituir per mitjà de la propietat 'arg0'

```
<arg0 key="nameForm.firstname.displayname"/>
```

Exemple:

```
<field property="firstname" depends="required">  
  <arg0 key="nameForm.firstname.displayname"/>  
</field>
```

- **Definir camps amb Expressió Regular**

*Aplicable a: Valors de tipus 'String'*

- 1) Definir el valor 'mask' en l'atribut 'depends'
- 2) Definir la variable 'mask'.

En el tag '<var-name>' especificar 'mask' i en '<var-value>' la màscara a utilitzar.

```
<var>  
  <var-name>mask</var-name>  
  <var-value>^[a-zA-Z]*$</var-value>  
</var>
```

- 3) Per defecte, el missatge que es mostrarà s'ha definit en la propietat 'msg' del tipus de validador (errors.mask)

```
<validator name="mask"  
  ...  
  msg="errors.mask"  
>
```

```
errors.mask=El campo {0} no cumple la máscara {1}
```

el fet que el missatge internacionalitzat requereix de 2 paràmetres podem especificar els valors per mitjà d'arguments del tipus 'arg0' i 'arg1'.



Generalment, el missatge que no es compleix una màscara no és adequat per als usuaris. Per exemple: El camp 'Nom' no compleix la màscara `'^[a-zA-Z]*$'` segurament deixa atònits els nostres usuaris. Per aquest motiu és millor definir un missatge adequat al camp en qüestió per mitjà de la propietat 'msg':

```
<msg name="mask" key="nameForm.errors.firstname.mask"/>
```

En cas de tenir que passar valors al nostre missatge definirem els arguments per mitjà de 'arg0',...'argN'.

Podem definir màscares globals en la secció constants:

```
<global>
  <constant>
    <constant-name>any</constant-name>
    <constant-value>^([0-9][0-9][0-9][0-9])$</constant-value>
  </constant>
</global>
```

i referenciar-les des de la variable '<var-value>' de la màscara per mitjà de '\$ {nombreConstante}'

```
<var>
<var-name>mask</var-name>
  <var-value>${any}</var-value>
</var>
```

- **Definir camps de tipus Data**

*Aplicable a: Valors de tipus 'String'*

- 1) Definir el valor 'date' en l'atribut 'depends'
- 2) Definir la variable 'datePattern'. Com a valor definir el patró tal com realitza la classe 'java.text.SimpleDateFormat'

```
<field property="birthDate" depends="date">

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

```
</field>
```

La validació de tipus data es realitza així en cas que s'hagi definit l'atribut com a String. Si l'atribut és de tipus 'Date' no té sentit realitzar la validació sobre aquest tipus de dades, doncs aquest ja és un tipus vàlid. En cas que el camp sigui de tipus 'java.util.Date' **no** s'ha de definir la validació al fitxer. Veure 'Integració amb el Servei de Presentació' per veure com realitzar la validació si l'objecte no és de tipus String.

NOTA: En cas d'utilitzar diferents formats de data (idiomes amb formats diferenciats) usar formsets. Veure 'Integració amb altres Serveis- Integració amb el Servei d'Internacionalització'.

- **Definir camps de tipus Numéric**

*Aplicable a valors de tipus 'String'*

Per comprovar si un valor entrat es podrà transformar en un valor numèric tenim diverses possibilitats segons el tipus de dades de destí al que anirà el valor entrat.

- 1) Definir en l'atribut 'depends' un dels següents valors segons el tipus de camp destí:

byte	Validació de que el camp pot ser convertit a Byte (entre $-2^7$ i $2^7-1$ )
short	Validació de que el camp pot ser convertit a Short (entre $-2^{15}$ i $2^{15}-1$ )
integer	Validació de que el camp pot ser convertit a Integer (entre $-2^{31}$ i $2^{31}-1$ )
long	Validació de que el camp pot ser convertit a Long (entre $-2^{63}$ i $2^{63}-1$ )
float	Validació de que el camp pot ser convertit a Float (entre $2^{-149}$ i $(2 \cdot 2^{-23}) \cdot 2^{127}$ )
double	Validació de que el camp pot ser convertit a Double (entre $2^{-1074}$ i $(2 \cdot 2^{-52}) \cdot 2^{1023}$ )

- 2) Definir a l'argument 0 la clau del camp del formulari o atribut

```
<arg0 key="order.prodno" />
```

- 3) En el fitxer 'validator-rules.xml' apareix quin és el missatge que es mostrarà a l'usuari en cas de validació incorrecta. Per defecte s'han de definir els missatges

```
errors.byte=...{0}....
```

```
errors.short=.... {0} ....  
errors.integer=.... {0} ....  
errors.long=.... {0} ....  
errors.float=.... {0} ....  
errors.double=.... {0} ....
```

- **Definir Longituds Mínimes i Màximes dels Camps**

*Aplicable a valors de tipus 'String'*

- 1) Definir el valor 'maxlength' i/o 'minlength' en l'atribut 'depends'
- 2) Definir a la variable 'maxlength' i/o 'minlength' els valors de longitud permesos

```
<var><var-name>maxlength</var-name><var-value>20</var-value></var>
```

- 3) Per defecte, el missatge que es mostrarà s'ha definit en la propietat 'msg' del tipus de validador

```
errors.minlength= El camp {0} no pot tenir menys de {1} caracters.
```

el fet que el missatge internacionalitzat requereix de 2 paràmetres podem especificar els valors per mitjà d'arguments del tipus 'arg0' i 'arg1'. Segons el missatge mostrat l'argument '0' correspondrà a la clau del camp, mentre que l'argument '1' seria el nombre de caràcters permesos (maxlength, minlength). Per especificar que volem fer servir com a argument 1 el valor de variable utilitzat podem fer servir la sintaxi \${var: nomVariable} i especificar el paràmetre 'resource=false' per indicar al validador que per aquesta clau el valor no s'ha d'obtenir del fitxer de recursos multiidioma, i per tant no s'ha de traduir.

```
<arg0 key="form.accountForm.name"/>  
<arg1 name="minlength" key="{var:minlength}" resource="false"/>
```

## 2.3. Utilització del Servei

La utilització del Servei es basa principalment en la configuració. L'ús directe des dels clients es permet mitjançant les interfícies definides.

openFrame ofereix ja la integració des de 2 punts diferenciats:



- Des del servidor, segons el 'pojoClass' configurat a l'acció (veure document 'Serveis de Presentació' i 'Servei de Tags' per a més referència)
- Des del client mitjançant la generació automàtica de Javascript de validació (veure el document 'Servei de Tags' per a més referència)

En l'apartat 'Integració amb Altres Serveis' es dona detall de com usar el Servei des de la capa de presentació, tot i que és necessària una lectura prèvia dels documents 'Serveis de Presentació' i 'Servei de Tags'.

## 2.4. Eines de Suport

L'expressió regular de les màscares definida a les validacions de tipus 'mask' es basa en PERL5. Si volem comprovar que s'ha definit correctament una expressió regular podem realitzar proves des de la pàgina '<http://jakarta.apache.org/oro/demo.html>'.

The screenshot shows a web-based Perl5 Expression Validator. At the top, there is a label 'Perl5 Expression:' followed by a dropdown menu. Below this, there is a text input field containing 'contains()'. To the right of the input field are two more dropdown menus: one for 'Case Sensitive' and another for 'Search'. Further right are two buttons: 'Search' and 'Reset'. Below these controls is a large text area labeled 'Search Input'. Underneath the input area is a section labeled 'Search Results', which contains a large, empty beige rectangular area for displaying the results of the validation.

## 2.5. Integració amb Altres Serveis

### 2.5.1. Integració amb el Servei de Internacionalització

En els fitxers de configuració es defineixen claus que permeten especificar quins missatges retornar en cas de validació errònia. Per a poder traduir aquestes claus és necessari especificar el Servei de Validació que usará el Servei d'Internacionalització (veure Configuració).

En determinats casos podem requerir de diferents tipus validació segons l'idioma seleccionat (dates, formats de moneda, etc.). En aquest cas, crearem diversos formsets..



Si fins ara hem definit el nostre conjunt de validacions dins del tag '<formset>', podem agrupar diferents conjunts de validacions segons l'idioma per mitjà del format següent:

```
<formset language="xx">  
</formset>
```

, on 'xx' representa la codificació ISO del llenguatge segons l'estàndard especificat en:

<http://ftp.ics.uci.edu/pub/ietf/http/related/iso639.txt>

De forma opcional podem especificar el país dins del llenguatge, la sintaxi seria:

```
<formset language="xx" country="yy">  
</formset>
```

, on 'yy' representa la codificació ISO del país segons l'estàndard especificat en:

[http://userpage.chemie.fu-berlin.de/diverse/doc/ISO\\_3166.html](http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html)

### **2.5.2. Integració amb el Servei de Presentació**

El Servei de Presentació defineix la possibilitat de 2 tipus de validació: des de Client o Servidor. El tipus de validació que es farà servir es defineix al tag de formulari (consultar 'Servei de Tags – Tag Form').

La diferència entre els 2 mecanismes és que en el cas de Client la validació es fa des del navegador del Client (mitjançant javascript) i en el cas de Servidor es fa servir Ajax per comunicar des del navegador amb el servidor i presentar els missatges sense haver de recarregar de nou la pàgina.

En cas que la validació es faci per mitjà de Javascript incrustat a la pàgina, no cal realitzar cap procés adicional. El tag formulari s'encarrega de generar el codi de validació Javascript basat en el validador configurat.

Si fem servir la validació des de Servidor amb Ajax, openFrame proporciona una classe de validació des de Web que permet validar paràmetres de tipus Map. En aquest servei de validació, adicionalment es realitza el binding entre els paràmetres rebuts i l'objecte de destí.

Per configurar aquesta validació realitzarem els següents passos:

- 1) Definir el servei de validació web

***Fitxers de configuració: openFrame-services-web.xml***

*Ubicació proposada: <PROJECT\_ROOT>/src/main/resources/spring*



Cal que es referenciïn els serveis d'internacionalització i logging, així com la factoria de validació ja definida pel Servei de Validació.

```
<bean id="webValidationService"
class="net.opentrends.openframe.services.web.validation.commons.CommonsWebValidati
onServiceImpl">

    <property name="webDataBinderFactory">
        <ref bean="webDataBinderFactory" />
    </property>

    <property name="validatorFactory">
        <ref bean="validatorFactory" />
    </property>

    <property name="i18nService">
        <ref bean="i18nService" />
    </property>

    <property name="loggingService">
        <ref bean="loggingService" />
    </property>

</bean>
```

- 2) Definir el 'webDataBinderFactory'. Aquesta factoria (net.opentrends.openframe.services.web.spring.bind.WebDataBinderFactory') serà usada pel servei per passar dels paràmetres rebuts als atributs destí.

***Fitxers de configuració: openFrame-services-web.xml***

*Ubicació proposada:* <PROJECT\_ROOT>/src/main/resources/spring

```
<!-- Use init method to initialize in 'init-method' -->
<bean id="webDataBinderFactory"
class="net.opentrends.openframe.services.web.spring.bind.WebDataBinderFactory">
    <property name="customEditors"><ref bean="customEditors"/></property>
</bean>
```

La propietat 'customEditors' permet definir com tractar cadascun dels tipus rebuts.

- 3) Per cada tipus de destí al que hem de copiar els paràmetres podem definir editors especials que realitzen el pas. Per defecte existeixen ja varis editors configurats per fer el pas entre tipus de tipus bàsic.

***Fitxer de configuració: property-editors.xml***

*Ubicació proposada:* <PROJECT\_ROOT>/src/main/resources/spring

En el cas de openFrame s'ofereix un nou editor 'CustomDateEditor' que permet configurar com ha de ser la transformació de un String a Date. Per això permet configurar per cada llenguatge quin és el patró de data (basat en 'SimpleDateFormat') que ha de complir el paràmetre d'entrada per a poder fer la transformació.

```
<bean id="customEditors" class="java.util.HashMap">
  <constructor-arg>
    <map>
      <entry key="java.util.Date">
        <bean
class="net.opentrends.openframe.services.il8n.spring.beans.propertyeditors.CustomD
ateEditor">
          <property name="il8nService" ref="il8nService"/>
          <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>
      </entry>
    </map>
  </constructor-arg>
</bean>
```

El missatge d'error associat que es mostrarà serà el que es defineixi al fitxer de recursos amb clau:

'typeMismatch.nomAtribut'

Així, si volguessim mostrar l'error de validació de data per l'atribut 'birthDate', definiríem:

```
typeMismatch.birthDate=El camp 'Data de naixement' no és una data correcta
```

- 4) Una vegada definit el servei, hem de fer-lo accessible des d'Ajax. openFrame fa ús de la llibreria DWR d'Ajax per a comunicar-se amb els objectes publicats en Spring. Així doncs publicarem el nostre servei de validació com a objecte Javascript en el fitxer 'resources/dwr/dwr.xml'.

```
<dwr>
  <allow>
    ..
    <create creator="spring" javascript="validationService">
      <param name="beanName" value="validationService"/>
    </create>
    ...
  </allow>
</dwr>
```

Una vegada realitzats els anteriors passos, openFrame proporciona al fitxer 'src/main/webapp/scripts/ajax/ajaxtags/openFrame-ajaxtags-validation.js' el mecanisme de comunicació amb el servei. En la generació de la pàgina s'afegeix com a listener del submit del formulari i realitza la crida al servidor per a comprovar les validacions. Si tot és correcte farà el submit, en cas contrari presentarà els missatges d'error.

Per a presentar els missatges d'error requereix que es defineixi en alguna part de la pàgina:

- 1) Un 'div' amb id 'errorsZone' (que mostrarà o ocultarà segons hi hagin o no errors).
- 2) Un '<ul id="errors"></ul>'. Dins aquest codi incorporarà de forma dinàmica la llista d'errors. A més, openFrame incorporarà per cada error un link que en ser clickat ubicarà el focus al camp afectat per l'error.
- 3) Opcionalment es pot definir un '<span id="errorCount"></span>' per a que incorpori de forma dinàmica el nombre d'errors trobats

Un exemple de pàgina seria la següent:

```
<div class="error" id="errorsZone"
  style="margin-right: 10px; margin-bottom: 3px; margin-top: 3px; display:none">
  

  Se han encontrado <span id="errorCount"></span> errores en tu
formulario.
  <p>Por favor corrige estos errores y vuelve a
enviar el formulario:</p>
  <ul id="errorList"></ul>
</div>
```

NOTA: Podem substituir amb el tag 'fmt:message' de JSTL el missatge 'Se han encontrado...'


Si definim els següents estils per mostrar els resultats:





```
div.error, div.message {  
  background: #ffc;  
  border: 1px solid #000;  
  color: #000000;  
  font-family: Arial, Helvetica, sans-serif;  
  font-size: 1em;  
  font-weight: normal;  
  margin: 10px auto;  
  padding: 3px;  
  text-align: left;  
  vertical-align: bottom;  
  margin-right: 200px;  
}
```

El resultat obtingut seria el següent:

 Se han encontrado 2 errores en tu formulario.

Por favor corrige estos errores y vuelve a enviar el formulario:

- El campo Apellido es obligatorio
- Nombre no es del formato esperado

## 2.6. Preguntes Freqüents



### **3. Exemples**



## **4. Annexos**