

Bienvenido

Nombre de usuario

Contraseña

Iniciar sesión

[Olvidó su contraseña?](#)

[Nuevo usuario?](#)

[Regístrese aquí](#)

Validaciones en JSF

Escrito por fetishcode

martes, 17 octubre 2006

Objetivo:

Este artículo tratara de explicar brevemente los diferentes tipos de validación que podemos utilizar en JSF.

El ejemplo se basará en un formulario en el cual se validaran campos de distintas maneras y desde distintos sitios de la aplicación

A continuación podemos ver el formulario de prueba creado para llevar a cabo este artículo.

Formulario

* CustomerId	<input type="text"/>
* CustFirstName	<input type="text"/>
* CustLastName	<input type="text"/>
CustStreetAddress1	<input type="text"/>
CustStreetAddress2	<input type="text"/>
CustCity	<input type="text"/>
CustState	<input type="text"/>
CustPostalCode	<input type="text"/>
PhoneNumber1	<input type="text"/>
PhoneNumber2	<input type="text"/>
CreditLimit	<input type="text"/>
* CustEmail	<input type="text"/>

Figura1

Requisitos:

-JDeveloper 10g (10.1.3).

-Una base de datos, para la realización de este articulo se ha hecho uso de Oracle 10g Express Edition

Encuesta

¿Ya está utilizando JSF o ADF Faces en sus proyectos ?

SI

No, todavía uso JSP puro.

No, todavía uso JSP con Struts.

No, todavía uso JSP, Struts, JSTL y Binding.

Ultimos Artículos

JSF - Lifecycle **Nuevo!**

JSF - Jerarquía Tree **Nuevo!**

JSF - Internacionalización **Nuevo!**

ProyectoWeb, Jdeveloper y JasperReport

JSF - CREANDO UN LOGIN

Componente Tree de UIX

Proyecto Web - Parte II (Business Layer)

Artículos Destacados

Curso de ADF BC, JSP, Struts con Jdeveloper 10.1.2

ProyectoWeb, JDeveloper y JasperReport

JSF - Creando un login Proyecto Web - Parte II (BUSINESS LAYER)

Introducción a Business Component - Parte I

Buscar Artículos

Usuarios Registrados

 Registrados: 3898

1. Tipos de Validaciones

El validar los datos de un formulario es una tarea ya común en cualquier aplicación web, pero no existe un estándar definido de cómo y donde validar los datos introducidos por el usuario.

Podríamos agrupar las validaciones de (JSF, ADF BC) en cuatro tipos bien diferenciados:

- Validaciones estándar
- Validaciones en el backing bean
- Validaciones en la capa de negocio, a nivel de aplicación
- Validaciones con nuestro propio Validator

Antes de empezar a explicar brevemente cada una de las validaciones hay que destacar que trabajamos con ADF, un framework de desarrollo que una de las ventajas que nos ofrece es poder ahorrar tiempo y simplificar las tareas en tiempo de desarrollo.

El campo de las validaciones es un claro ejemplo, al crear un formulario podremos observar como ADF por si solo nos ha implementado todo un conjunto de validaciones básicas como son:

-Controlar que los campos que a nivel de base de datos son "NOT NULL", o a nivel de Entity han sido marcados como "Mandatory" sean rellenos.

-Controlar que en los campos que a nivel de base de datos son de tipo "NUMBER" no se introduzcan caracteres.

1.1: Validaciones estándar

Este tipo de validaciones son las mas básicas, y se realizan a nivel de view, en el jsp.

Ejemplos de este tipo de validaciones son comprobar que un valor este entre un determinado rango, comprobar la longitud del valor introducido, etc.

Este tipo de validaciones las podemos encontrar en la *Component Palette* del Jdeveloper, pestaña *JSF Core: Validate Length, Validate DoubleRange, Validate LongRange*

A continuación validaremos el campo *CustomerId* con un *Validate LongRange*

```
<af:inputText value="#{bindings.CustomerId.inputValue}"
  label="#{bindings.CustomerId.label}"
  required="#{bindings.CustomerId.mandatory}"
  columns="#{bindings.CustomerId.displayWidth}"
  binding="#{backing_pantallas_Form.inputText1}"
  id="inputText1" autoSubmit="true">
  <af:validator binding="#{bindings.CustomerId.validator}"/>
  <f:convertNumber groupingUsed="false"
    pattern="#{bindings.CustomerId.format}"/>
  <f:validateLongRange minimum="3" maximum="14" />
</af:inputText>
```

En la figura2 podemos ver como nos salta el mensaje en forma de pop up al introducir como *CustomerId* el numero 15, el cual se encuentra fuera del rango marcado.

También nos indica que hay una serie de campos que deben ser rellenos, ya que no pueden ser nulos, ya sea debido a una restricción a nivel de base de datos o a nivel de Entity impuesta por nosotros.

usuarios.

 Este mes: 55 usuarios.

RSS Disponible

RSS	0.91
RSS	1.0
RSS	2.0
ATOM	0.3

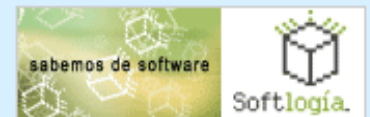
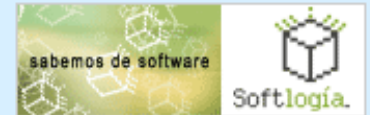


Figura2

1.2: Validaciones en el backing bean

Este tipo de validaciones suelen ser un poco más complejas que las anteriores. Y al realizarse en el backing pueden tratarse de validaciones múltiples que tengan en cuenta valores de otros campos del formulario.

A continuación podemos ver que al campo CustFirstName se le ha añadido un validator que se encuentra en el backing

```
<af:inputText value="#{bindings.CustFirstName.inputValue}"
  label="#{bindings.CustFirstName.label}"
  required="#{bindings.CustFirstName.mandatory}"
  columns="#{bindings.CustFirstName.displayWidth}"
  binding="#{backing_pantallas_Form.inputText2}"
  id="inputText2" autoSubmit="true" immediate="true"
  validator="#{backing_pantallas_Form.validaCampo}">
</af:inputText>
```

El método que deberíamos tener en el *backing* recibirá por parámetro el FacesContext, el componente que ha provocado el evento de validación, y el valor que tiene dicho componente.

```
public void validaCampo(FacesContext facesContext, UIComponent uiComponent,
  Object object) {
  //validar el campo y actuar en consecuencia
}
```

1.3: Validaciones en la capa de negocio.

Este tipo de validaciones tratan de validar un proceso de nuestra aplicación, se trata de validar el proceso que se realizará al rellenar el formulario.

Es decir, en este tipo de validaciones más que validar que los datos introducidos sean correctos, lo que se comprueba es que se pueda hacer una determinada tarea con esos datos.

Imaginemos el siguiente caso, un cliente desea realizar una determinada gestión y para ello debe rellenar un formulario. Una vez rellenado el formulario y validados sus datos la aplicación recoge los datos necesarios de dicho cliente para verificar si tiene gestiones aun pendientes. Este tipo de validaciones deberíamos implementarlas en la capa de negocio, ya sea nivel de ViewObject o a nivel de Entity según sea necesario.

1.4: Validaciones con nuestro propio Validator

Este tipo de validadores es frecuente usarlos cuando tenemos un tipo de campo que hay que validar y que se repite con frecuencia en diferentes formularios de nuestra aplicación, como podría ser validar un NIF o un número de cuenta bancaria, etc.

Ante esta situación es más eficiente crearnos nuestro propio Validator y no tener que implementar lo mismo en diversos backings

Para ello lo que haremos es crear nuestro propio Validator. Como podemos observar el método validate recibe por parámetro lo mismo que nuestro método de validación en el backing bean

```
package view.utils;

import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.faces.component.UIComponent;
import javax.faces.component.UIInput;

public class MyValidator implements Validator{

    public void validate(FacesContext context, UIComponent comp, Object valor){

        //aquí validamos el campo

    }

}
```

Seguidamente debemos registrar nuestro validator en el *faces_config.xml*. Esto lo podremos hacer desde el *Source* o mediante wizard desde *Overview*

```
<validator>
  <validator-id>Myvalidator</validator-id>
  <validator-class>view.utils.MyValidator</validator-class>
</validator>
```

Finalmente lo último que tenemos que hacer es asignar nuestro validator al campo del formulario.

```
<af:inputText value="#{bindings.CustLastName.inputValue}"
  label="#{bindings.CustLastName.label}"
  required="#{bindings.CustLastName.mandatory}"
  columns="#{bindings.CustLastName.displayWidth}"
  binding="#{backing_pantallas_Form.inputText3}"
  id="inputText3" autoSubmit="true" immediate="true" >
  <af:validator binding="#{MyValidator}" validatorId="Myvalidator"/>
</af:inputText>
```

Conclusión:

Ahora ya conocemos los distintos sitios y formas de validar nuestros campos en JSF, y podremos dependiendo de nuestro entorno y requisitos implementar la forma que mas nos convenga. Seguramente si implementamos una gran aplicación acabamos haciendo uso de casi todas.

Acerca del autor

Alejandro Font es Ingeniero Técnico de Telecomunicaciones especializado en Telemática. Actualmente trabaja como desarrollador de aplicaciones J2EE, tanto ADF/Swing como ADF/ JSF (Java Server Faces) y es uno de los componentes del equipo de <http://fetishcode.bitacorras.com/>

Comentario[s]

Sólo los usuarios registrados pueden escribir comentarios.

Por favor identificate o regístrate.

Powered by [AkoComment 2.0!](#)

[< Anterior](#)

[Siguiente >](#)

[\[Atrás \]](#)

© 2008 JDeveloperLA.com - La comunidad en español de JDeveloper
[Joomla!](#) is Free Software released under the GNU/GPL License.