PORTIONS (PORTlet actIONS)

Guía de Usuario

Controlador

Especificación del controlador de la aplicación

portlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-
app_1_0.xsd"
        version="1.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet
-app_1_0.xsd">
    <portlet>
        <description>
            Descripción del portlet de ejemplo
        </description>
        <portlet-name>
            PortletDeEjemplo
        </portlet-name>
        <display-name>
            Portlet de Ejemplo
        </display-name>
        <portlet-class>
            net.sf.portions.controller.PortletController
        </portlet-class>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>VIEW</portlet-mode>
            <portlet-mode>EDIT</portlet-mode>
            <portlet-mode>HELP</portlet-mode>
        </supports>
        <supported-locale>en</supported-locale>
        <portlet-info>
            <title>Portlet de Ejemplo</title>
            <short-title>PortletDeEjemplo</short-title>
            <keywords>Palabras clave</keywords>
        </portlet-info>
    </portlet>
</portlet-app>
```

Configuración del controlador

portlet-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-config>
    <form-beans>
        <form-bean name="sampleForm"
                type="net.sf.portions.sample.form.SamplePortletForm"
/>
```

```
        ...
    </form-beans>
    <global-forwards>
        <forward name="success"
                mode="view"
                path="/jsp/view_1.jsp"
                content="text/html"/>
        <forward name="failure"
                mode="view"
                path="/jsp/view_1.jsp"
                content="text/html"/>
        ...
    </global-forwards>
    <action-mappings>
        <action name="step1"
                type="net.sf.portions.sample.action.Step1PortletAction
">
            <forward name="success"
                    mode="view"
                    path="/jsp/view_1.jsp"
                    content="text/html"/>
        </action>
        <action name="step2"
                type="net.sf.portions.sample.action.Step2PortletAction
"
                input="/jsp/view_1.jsp"
                content="text/html"
                formName="sampleForm">
            <forward name="success"
                    mode="view"
                    path="/jsp/view_2.jsp"
                    content="text/html"/>
        </action>
        ...
    </action-mappings>
    <message-resources parameter="resources.ApplicationResources" />
    <plug-in className="net.sf.portions.validator.ValidatorPlugIn">
        <set-property property="pathnames" value="/WEB-
INF/validation.xml"/>
    </plug-in>
</portlet-config>
```

Existen cinco secciones diferenciadas dentro del fichero de configuración portlet-config.xml:

- **elemento "form-beans":** lista los diferentes formularios de la aplicación. Por cada formulario se definirá elemento "form-bean" con los siguientes atributos:
    - "name": identificador del formulario
    - "type": clase que implementa el formulario
- **elemento "global-forwards":** identifica las redirecciones (forwards) globales (por defecto) de la aplicación. Por cada redirección se definirá un elemento "forward" con los atributos:
    - "name": identificador del forward
    - "mode": modo del portlet (edit, help o view)
    - "path": url a la página JSP a la que se realizará la redirección
    - "content": content-type generado

- **elemento "action-mappings":** mapea cada una de las actions de la aplicación utilizando elementos "action", que podrán tener los siguientes parámetros:
    - "name": identificador de la action
    - "type": clase que implementa la action
    - "input": url a la página JSP a la que se redireccionará al producido errores durante la ejecución de los plug-ins
    - "content": content-type de la página JSP de error indicada en el parámetro "input"
    - "formName": nombre del formulario asociado a la action

  Además, cada elemento "action" contendrá tantos elementos "forward" (iguales que los definidos en la sección "global-forwards") como posibles redirecciones genere la ejecución de la action.

- **elemento "message-resources":** especifica, a través del atributo "parameter", el archivo de properties con los recursos y mensajes propios de la aplicación.
- **lista de elementos "plug-in":** permite listar los plug-ins (por ejemplo el plug-in de validación) que debe ejecutar el controlador. Por cada plug-in se especificará un elemento "plug-in", con el siguiente parámetro:
    - "className": indicará la clase que implementa el plug-in

  Si es necesario se podrán especificar propiedades de configuración para cada plug-in con los sub-elementos "set-property", que a su vez tienen los siguientes parámetros:

    - "property": identifica la propiedad a configurar
    - "value": especifica el valor para dicha propiedad

Las clases PortletController y ConfigHelper

El controlador, dentro del patrón MVC, es el encargado del procesamiento de las peticiones de entrada y la creación de cualquier bean u objeto usado por las páginas JSP, así como de la toma de decisiones, dependiendo de las acciones del usuario, de a qué página JSP debe redireccionarse la salida. Estas tareas, dentro del framework PORTIONS, son llevadas a cabo por la clase PortletController.

La clase ConfigHelper permite obtener los datos de configuración de la aplicación indicados a través del fichero "portlet-config.xml".

PortletController

Propiedades

log

Log de la clase

Métodos

init(PortletConfig)

Realiza las tareas de inicialización del controlador, que incluyen el realizar una llamada al método init de la clase ConfigHelper para cargar los datos de configuración del fichero /WEB-INF/portlet-config.xml.

doView(Forward, RenderRequest, RenderResponse)

Renderiza el modo view del portlet.

doEdit(Forward, RenderRequest, RenderResponse)

Renderiza el modo edit del portlet.

doHelp(Forward, RenderRequest, RenderResponse)

Renderiza el modo help del portlet.

processAction(ActionRequest, ActionResponse)

Procesa una petición de acción al portlet.

getActualForward(PortletRequest):Forward

Permite obtener el forward actual al que el controlador ha de pasar el control.

ConfigHelper

Métodos

init(URL)

Carga los datos de configuración a partir de la URL al fichero portlet-config.xml.

getAction(String):Action

Retorna los datos de configuración de la action cuyo nombre se pasa como parámetro.

getActionType(String):String

Retorna el valor de la propiedad type de la action cuyo nombre se pasa como parámetro.

getActionInput(String):String

Retorna el valor de la propiedad input de la action cuyo nombre se pasa como parámetro.

getActionContent(String):String

Retorna el valor de la propiedad content de la action cuyo nombre se pasa como parámetro.

getActionScope(String):String

Retorna el valor de la propiedad scope de la action cuyo nombre se pasa como parámetro.

getActionFormName(String):String

Retorna el valor de la propiedad formName de la action cuyo nombre se pasa como parámetro.

getActionParameter(String):String

Retorna el valor de la propiedad parameter de la action cuyo nombre se pasa como parámetro.

getActionForwards(String):Map

Retorna un Map que contiene los forwards asociados a la action cuyo nombre se pasa como parámetro.

getForward(String, String, String):Forward

Retorna los datos de configuración del forward identificado a través de los parámetros pasados.

getForwardPath(String, String, String):String

Retorna el valor de la propiedad path del forward identificado a través de los parámetros pasados.

getForwardContent(String, String, String):String

Retorna el valor de la propiedad content del forward identificado a través de los parámetros pasados.

getForm(String):FormBean

Retorna los datos de configuración del form cuyo nombre se pasa como parámetro.

getFormType(String):String

Retorna el valor de la propiedad type del form cuyo nombre se pasa como parámetro.

getDefaultForward(String, String):Forward

Retorna los datos de configuración de un forward global (por defecto) identificado a través de los parámetros pasados.

getDefaultForwardPath(String, String):String

Retorna el valor de la propiedad path de un forward global (por defecto) identificado a través de los parámetros pasados.

getDefaultForwardContent(String, String):String

Retorna el valor de la propiedad content de un forward global (por defecto) identificado a través de los parámetros pasados.

getMessageResources():String

Retorna los datos de configuración correspondientes a la ubicación del fichero MessageResources.

getPluginNames():List

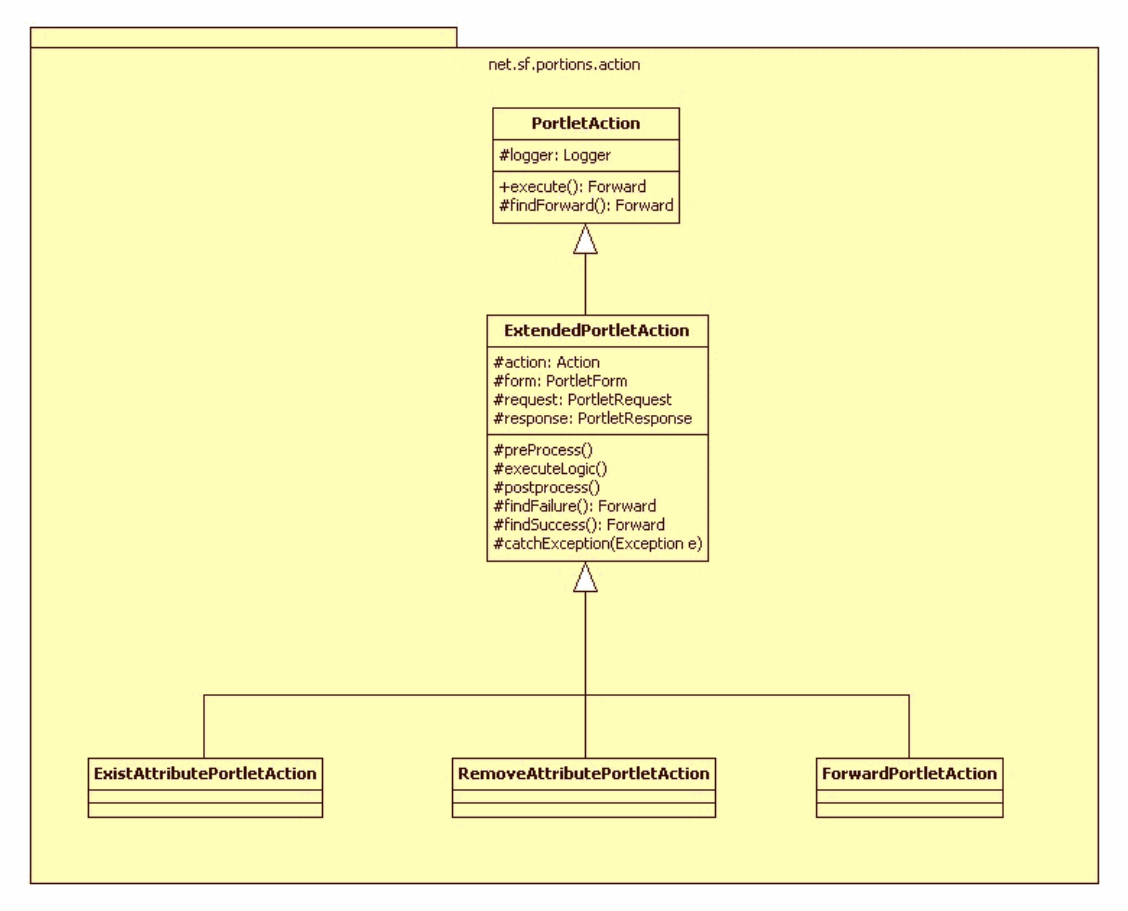
Retorna una lista de los nombres de los plug-in's configurados.

getPluginParam(String, String):String

Retorna el valor del parámetro del plug-in identificado a través de los parámetros pasados.

Jerarquía de Actions

A continuación se muestra un Diagrama de Clases que representa la jerarquía de clases Action proporcionada por el framework PORTIONS:



- Figura 1. Jerarquía de Actions del framework PORTIONS -

PortletAction

Clase base de la jerarquía de Actions proporcionada por el framework PORTIONS. Se trata de una clase abstracta que ofrece la funcionalidad básica que utilizarán las Actions.

Propiedades

log

Log de la clase.

Métodos

execute(Action , PortletForm , PortletRequest , PortletResponse):Forward

Método abstracto que deberán sobreescribir las Actions concretas para ejecutar sus tareas.

éste método será invocado por el controlador de la aplicación.

findForward(Action , String , String): Forward

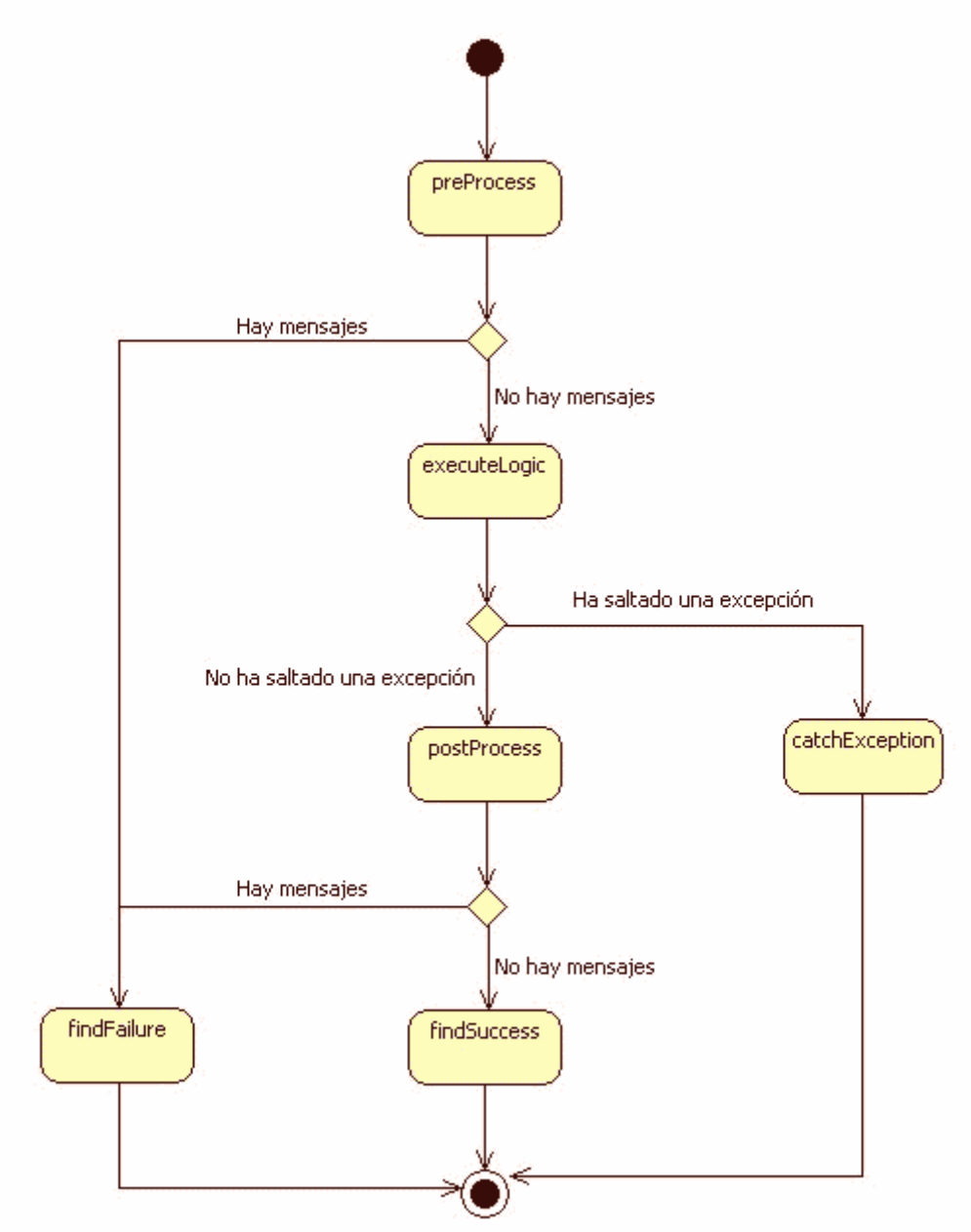
Método que permite obtener el forward al que se redireccionará el flujo de la aplicación.

La implementación por defecto extrae el forward de la instancia de la clase Action que se pasa por parámetro, y si no existe, busca el forward por defecto.

Las clases hijas pueden sobreescribir este método.

ExtendedPortletAction

Clase abstracta que representa una extensión de la clase base de la jerarquía de Actions. Implementa la siguiente máquina de estados:



- Figura 2. Máquina de Estados de la clase ExtendedPortletAction -

Propiedades

action

Instancia de la clase net.sf.portions.controller.configuration.Action con los datos de configuración de la action.

form

Instancia de la clase net.sf.portions.form.PortletForm que opcionalmente (dependiendo de la configuración) puede contener los parámetros de la petición.

request

La petición procesada (instancia de la clase javax.portlet.PortletRequest).

response

La respuesta generada (instancia de la clase javax.portlet.PortletResponse).

Métodos

preProcess()

Comprueba las precondiciones que debe cumplir la Action.

En el caso de que no se cumpla alguna precondición se debe dejar un registro de la incidencia mediante la creación de un mensaje. Esta acción provocará la redirección del flujo de ejecución hacia una página de error mediante la invocación del método findFailure().

executeLogic()

Ejecuta la lógica de negocio de la Action.

éste método debe ser sobrescrito de forma obligatoria por aquellas Actions que hereden de la clase ExtendedPortletAction.

postProcess()

Comprueba las postcondiciones que debe cumplir la Action.

En el caso de que no se cumpla alguna precondición se debe dejar un registro de la incidencia mediante la creación de un mensaje. Esta acción provocará la redirección del flujo de ejecución hacia una página de error mediante la invocación del método findFailure().

findFailure():Forward

Retorna una instancia de la clase Forward que redirecciona a una página de error.

La implementación por defecto obtendrá el Forward "failure" del modo (view, edit o help) que corresponda.

findSuccess():Forward

Retorna una instancia de la clase Forward que redirecciona a una página de éxito.

La implementación por defecto obtendrá el Forward "success" del modo (view, edit o help) que corresponda.

catchException(Exception)

Encierra el tratamiento de cualquier excepción que se pueda lanzar durante la ejecución de la lógica de negocio de la Action.

La implementación por defecto del método se limita a lanzar una excepción del tipo ExecuteActionException cuyo mensaje se corresponderá con el mensaje de la excepción capturada.

ExistAttributePortletAction

Verifica la existencia de un atributo en alguno de los scopes o ámbitos posibles (request, session) del portlet. Si el atributo existe se pasará el control a un forward etiquetado con "success" y, sino, a uno etiquetado con "failure".

En la propiedad "parameter" del <action-mapping> correspondiente se indicará el scope (ámbito) y el nombre del atributo a buscar, separados por ";" (punto y coma):

```
parameter="application;HOURS"
```

Si se quiere buscar el atributo en cualquiera de los posibles ámbitos se debe utilizar un "*" (asterisco):

```
parameter="*;HOURS"
```

RemoveAttributePortletAction

Elimina un atributo de uno de los scopes o ámbitos posibles (request, session) del portlet. Si el atributo existe y puede ser borrado se pasará el control a un forward etiquetado con "success" y, sino, a uno etiquetado con "failure".

En la propiedad "parameter" del <action-mapping> correspondiente se indicará el scope y el nombre del atributo a eliminar, separados por ";" (punto y coma):

```
parameter="application;HOURS"
```

Si se quiere eliminar el atributo en cualquiera de los posibles ámbitos (en este caso, el atributo sólo será eliminado del primer contexto en el que sea localizado) se utilizará un "*" (asterisco):

```
parameter="*;HOURS"
```

ForwardPortletAction

Permite pasar de una página JSP a otra siguiendo el flujo normal de la arquitectura (Modelo 2 de Arquitectura JSP aplicado a los portlets JSR-168). Para ello sólo es necesario configurar en el <action-mapping> correspondiente un forward "success" a la página JSP a la que se redireccionará.

Forms

Los "Forms" son javaBeans que opcionalmente se asociarán a una petición. En caso de que se defina en el fichero de configuración portlet-config.xml, dicho bean tendrá sus propiedades inicializadas con los parámetros correspondientes de la petición antes de que se ejecute la Action asociada a ésta.

PortletForm

Clase base de la jerarquía de Forms. Se trata de una clase abstracta que ofrece la funcionalidad básica que utilizarán los Forms.

Propiedades

log

> Log de la clase.

Métodos

validate()

> Realiza validaciones sobre los datos cargados en las propiedades.
>
> Actualmente sin usar por parte del controlador, se recomienda no utilizar.

reset()

> Resetea los valores de las propiedades.

set(ValueObject)

> Establece los valores de las propiedades de la instancia actual a partir de los datos obtenidos de un ValueObject.

populate():ValueObject

> Retorna un objeto ValueObject cargado con los datos de la instancia actual.

Vista

Librería de tags "portions"

messages

<portions:messages>: muestra los mensajes generados por la aplicación (normalmente errores de validación, o errores producidos durante la ejecución).

Atributos

property

> Propiedad a la que estan asociados los mensajes que se quieren mostrar. Por defecto se mostrarán todos los mensajes, sin distinguir la propiedad a la que están asociados.

id

> Identifica los valores de cabecera, prefijo, sufijo y pie a usar para mostrar los mensajes. Por defecto se mostrarán los determinados por las propiedades "portions.messages.header", "portions.messages.prefix", "portions.messages.sufix", "portions.messages.footer" del fichero MessageResources; en caso de especificar un valor, a dichas propiedades se les concatenará un "." (punto) seguido del identificador indicado.

```
<%@ taglib uri='http://java.sun.com/portlet' prefix='portlet'%>
<%@ taglib uri="/WEB-INF/tld/portions.tld" prefix="portions" %>
<portlet:defineObjects/>
<portions:messages />
```

getResource

<portions:getResource>: obtiene el valor asociado a una clave del fichero MessageResources.

Atributos

key

> Clave del fichero MessageResources.

```
<%@ taglib uri='http://java.sun.com/portlet' prefix='portlet'%>
<%@ taglib uri="/WEB-INF/tld/portions.tld" prefix="portions" %>
<portlet:defineObjects/>
<portions:getResource key="label.holaMundo" />
```

redirect

<portions:redirect>: realiza una redirección automática desde una JSP a una action del portlet.

Atributos

id

Identifica los valores del identificador, y del valor y la clase css del botón de envío del formulario que permite realizar la redirección. Por defecto se mostrarán los determinados por las propiedades "portions.redirect.formId", "portions.redirect.submitValue" y "portions.redirect.submitClass" del fichero MessageResources; en caso de especificar un valor, a dichas propiedades se les concatenará un "." (punto) seguido del identificador indicado.

action

Indica el nombre de la action a ejecutar.

mode

Especifica el modo del portlet a mostrar (help, edit o view).

```
<%@ taglib uri='http://java.sun.com/portlet' prefix='portlet'%>
<%@ taglib uri="/WEB-INF/tld/portions.tld" prefix="portions" %>
<portlet:defineObjects/>
<portions:redirect action="confirmacionAction" mode="view" />
```

encodeURL

<portions:encodeURL>: codifica una URL de un recurso estático (imagen, documento, ...) incluido en el WAR del portlet.

Atributos

url

url a codificar.

```
<%@ taglib uri='http://java.sun.com/portlet' prefix='portlet'%>
<%@ taglib uri="/WEB-INF/tld/portions.tld" prefix="portions" %>
<portlet:defineObjects/>
<img src="<portions:encodeURL url="img/imagenEjemplo.gif" />"
title="Imagen de ejemplo">
```

Validación de formularios

El framework PORTIONS incorpora un sistema de validación de formularios (basada en el proyecto Commons Validator) para el cual habrá que definir una serie de reglas de validación, y un mapeo de los campos de los formularios contra dichas reglas en uno o varios ficheros XML (comúnmente son los ficheros validation-rules.xml y validation.xml respectivamente, o simplemente el fichero validation.xml).

A continuación se muestra un ejemplo del fichero validation.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE form-validation PUBLIC
        "-//Apache Software Foundation//DTD Commons Validator Rules
Configuration 1.3.0//EN"
        "http://jakarta.apache.org/commons/dtds/validator_1_3_0.dtd">
```

```
<form-validation>
    <global>
        <validator name="required"
                  classname="net.sf.portions.validator.RequiredValidator
"
                  method="validateRequired"
                  methodParams="java.lang.Object,
org.apache.commons.validator.Field"
                  msg="error.required"/>

        <validator name="mustBeSelected"
                  classname="net.sf.portions.validator.RequiredValidator
"
                  method="validateRequired"
                  methodParams="java.lang.Object,
org.apache.commons.validator.Field"
                  msg="error.must_select"/>
    </global>

    <formset>
        <form name="patronForm">
            <field property="patron" depends="required">
                <arg key="patronForm.patron"/>
            </field>
        </form>

        <form name="idForm">
            <field property="id" depends="mustBeSelected">
                <arg key="idForm.id"/>
            </field>
        </form>
    </formset>
</form-validation>
```

En este ejemplo se observa como se han combinado en un único fichero las reglas de validación y el mapeo anteriormente comentados:

- Las reglas están contenidas en el elemento "global". Como se puede observar se han definido dos reglas diferentes usando el mismo validador, una denominada "required" y otra "mustBeSelected".
- Los mapeos entre los campos de los formularios y las reglas de validación se especifican dentro del elemento "formset". Cada formulario debe mapearse dentro de un elemento "form" que contendrá aquellas validaciones a realizar sobre cada campo en elementos "field".

Si se quiere profundizar en la validación de formularios se recomienda visitar la Web del proyecto Commons Validator.

Validadores incluidos en el framework PORTIONS

requerido

Clase: net.sf.portions.validator.RequiredValidator

Método: validateRequired

Valida la existencia de un campo del formulario y que dicho campo contenga datos.

Tablas de Paginación

Para la creación de tablas se hace uso de la librería DisplayTag (http://displaytag.sourceforge.net/). Esta librería permite construir una tabla HTML con funcionalidades de paginación, ordenación, etc; a partir de una colección de datos mediante un conjunto de etiquetas personalizadas.

Para compatibilizar el uso de la DisplayTag con el framework PORTIONS se incluyen en este último unas clases de soporte a esta librería. Estas clases se configurarán en el fichero displaytag.properties dando al parámetro factory.reqestHelper el valor siguiente:

```
factory.requestHelper=net.sf.portions.displaytag.PortletRequestHelperF
actory
```

Modelo

Objectos de datos

ValueObject *(Interface)*

El framework Portions ofrece el interface "ValueObject" que deben implementar los objetos de datos (patrón Value Object). Este interface define el método toXML() que permite ver una descripción del objeto en formato XML.

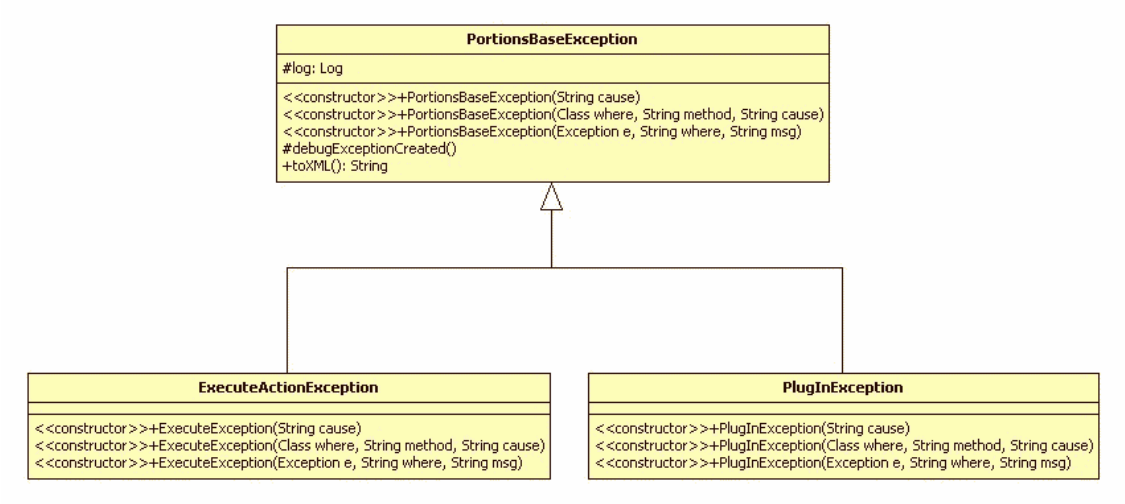
BaseVO

Para facilitar la implementación de los objetos de datos, el framework PORTIONS incluye la clase base "BaseVO" que realiza una implementación por defecto para el método toXML() definido en el interface ValueObject.

Otros

Jerarquía de Excepciones

A continuación se muestra un Diagrama de Clases que representa la jerarquía de Excepciones lanzadas por el framework PORTIONS:



- Figura 3. Jerarquía de Excepciones del framework PORTIONS -

PortionsBaseException

Clase base de la jerarquía de Excepciones proporcionada por el framework PORTIONS. Se trata de una clase que ofrece la funcionalidad básica que utilizarán las Excepciones del framework.

Propiedades

log

Log de la clase.

Métodos

<<constructor>>PortionsBaseException(String)

Constructor de la excepción que recibe como parámetro la descripción de la causa, el lugar o el motivo que provoca el disparo de la misma.

<<constructor>>PortionsBaseException(Class , String , String)

Constructor de la excepción que recibe como parámetros la clase donde se produce, el nombre del método donde se dispara y una descripción del motivo que provoca el disparo de la excepción.

<<constructor>>PortionsBaseException(Exception, String, String)

Constructor de la excepción que recibe como parámetros la excepción que provoca que se dispare esta excepción (excepción que se anidará), el lugar del código, operación o función que provoca el disparo de la excepción y el motivo, sugerencia o cualquier otro tipo de mensaje auxiliar de la excepción.

debugExceptionCreated()

Escribe información de depuración acerca de la excepción en el log.

toXML():String

Proporciona la representacion en XML de la Excepcion.

ExecuteActionException

La excepción ExecuteActionException será lanzada por una action si durante la ejecución de sus tareas se produce algún tipo de error o situación excepcional.

En caso de que una excepción de este tipo llegue el controlador, éste mostrará un mensaje de error en el log del portlet.

PlugInException

La excepción PlugInException será lanzada por un plug-in (clase que implementa el interface net.sf.portions.plugin.IPortletPlugIn) si durante la ejecución de sus tareas se produce algún tipo de error o situación excepcional que deba ser tratada por parte del controlador.

El controlador, al capturar una excepción de este tipo, obtendrá los mensajes que se hayan generado durante la ejecución del plug-in y redireccionará la ejecución a la página JSP indicada en el parámetro "input" de la action que se esté tratando.

Mensajes

Los mensajes son el medio a través de la cual se notificarán al usuario las causas de los errores (por ejemplo errores de validación) o las situaciones especiales que se produzcan durante la ejecución de las Actions.

El framework PORTIONS proporciona diferentes clases para el tratamiento de los mensajes (además del tag <portions:messages> ya comentado en la sección Vista):

MessageHelper

Clase de utilidad que proporciona métodos que permiten realizar tareas comunes relativas a los mensajes generados por la aplicación.

Métodos

saveMessages(PortletRequest, PortletMessages)

Almacena los mensajes pasados como parámetros en sesión.

setMessagesInRequest(PortletRequest)

Recupera y elimina de la sesión los mensajes almacenados para almacenarlos en la request.

thereAreMessages(PortletRequest)

Determina la existencia o la no existencia de mensajes en sesión.

PortletMessage

Clase que representa un mensaje generado por la aplicación. Representa una clave a un mensaje incluido en el fichero de MessageResources de la aplicación, y opcionalmente unos valores de reemplazo correspondientes a parámetros {0}, {1}, ... {n} de dicho mensaje.

Propiedades

key

Clave del mensaje (accesible a través del método getKey():String).

values

Valores de reemplazo del mensaje (accesible a través del metodo getValues():Object[]).

Métodos

<<constructor>>PortletMessage(String)

Constructor del mensaje que recibe como parámetro la clave correspondiente del fichero MessageResources de la aplicación.

<<constructor>>PortletMessage(String , Object[])

Constructor del mensaje que recibe como parámetro la clave correspondiente del fichero MessageResources de la aplicación y los valores de reemplazo.

getKey():String

Retorna el valor de la propiedad key.

getValues():Object[]

Retorna el valor de la propiedad values.

PortletMessages

Colección de mensajes con soporte para una jerarquización de mensajes por 'propiedad'.

Propiedades

GLOBAL_MESSAGE

Nombre de la propiedad utilizada para almacenar mensajes globales (es decir, aquellos no relacionados a una propiedad determinada).

Métodos

<<constructor>>PortletMessages()

Constructor sin parámetros de la clase.

<<constructor>>PortletMessages(PortletMessages)

Constructor de la clase que añade a la nueva instancia los mensajes contenidos en otra instancia pasada como parámetro.

add(String, PortletMessage)

Añade un mensaje al conjunto de mensajes asociado a laa propiedad determinada (si es null o cadena vacía se tomará el valor PortletMessages.GLOBAL_MESSAGE).

add(PortletMessages)

Añade todos los mensajes de una instancia de la clase PortletMessages al conjunto de mensajes actual.

clear()

limina todos los mensajes almacenados en la instancia actual.

isEmpty()

Determina si el conjunto actual de mensajes está vacío o no lo está.

get():Iterator

Retorna el conjunto de todos los mensajes almacenados.

get(String):Iterator

Retorna el conjunto de todos los mensajes almacenados asociados a la propiedad especificada (si es null o cadena vacía se tomará el valor PortletMessages.GLOBAL_MESSAGE).

properties():Iterator

Retorna un iterador con el conjunto de propiedades utilizados para almacenar mensajes.

size():int

Retorna el número de pares propiedad-clave añadidos.

size(String):int

Retorna el número de mensajes asociados a la propiedad especificada (si dicha propiedad es null, o cadena vacía, se tomará el valor PortletMessages.GLOBAL_MESSAGE).