Añadir Funcionalidad a OpenCms

1. Introducción

Aunque OpenCms con cada versión evoluciona más y mejor ofreciendo más funcionalidad que nos facilitan la creación de. En algunos casos pueden limitar nuestra creatividad y necesidades. Ante esta casuística tenemos dos posibilidades, bien integrar un módulo existente, ya que al tratarse de una herramienta de código libre hay muchos disponibles en la WWW; o bien crearte tu propio módulo.

2.Instalar módulos existentes

Por defecto OpenCms viene instalado con dos idiomas, Alemán e Inglés. Pero existe un módulo para traducir OpenCms 6.X al castellano que será el que instalaremos a modo de ejemplo.

Lo primero que debemos hacer es descargar el modulo denominado “OpenCms Locale ES” de la web de. A continuación nos situamos en la vista de administración de OpenCms y abrimos la opción *Module Management*, en ella nos saldrá un submenú en el que elegiremos *HTTP Import module*. Una vez dentro debemos examinar nuestro sistema para buscar el modulo descargado. Aceptamos y el modulo se habrá importado en nuestro OpenCms, aunque debemos reiniciar el entorno para que aparezca en castellano, para ello, desde la misma vista de administración vamos a *Workplace tool / Re-Initialize the Workplace*. Por último debemos escoger el idioma castellano en las preferencias de nuestro usuario.

3.Crear tu propio módulo

Lo primero que tenemos que hacer es crear un módulo desde la vista de administración, entrando en *Module Management.* Pulsamos sobre *New Module,* donde tan sólo debemos rellenar los campos en blanco. MUY IMPORTANTE: en *Package Name* debemos poner el mismo nombre que le pusimos al paquete java que hemos creado para este modulo, así mismo debemos marcar *create classes subfolder* y *create lib subfolder,* que creará una subcarpeta llamada *classes* y otra *lib.*

En *lib* meteremos nuestro jar, que previamente habremos creado con algún entorno de programación y que aportará las clases y métodos necesarios que nos darán la funcionalidad deseada. También pondremos todas las librerías que hayamos usado creando nuestras clases y que no vengan por defecto con OpenCms.

En la carpeta *classes* pondremos un fichero *properties* que contendrá las propiedades del módulo, e.g. la característica del idioma. El fichero debe tener la extensión *.properties*.

¿Qué es un fichero Properties?

Cuando desarrollamos una aplicación nos encontramos con constates y valores por defecto. Ponerlos directamente en el código es un gran error ya que cada modificación implicaría una recompilación del código. Un buen código no puede permitirse esos lujos y se hace imprescindible utilizar mecanismos que nos permitan modificar la configuración de nuestros programas de manera cómoda y efectiva. Por eso usamos el fichero properties, es decir, un fichero de configuración.

Un ejemplo este tipo de ficheros sería:

```
depuracion=True
alto=480
ancho=640
```

Para recuperar los valores hay dos maneras de hacerlo: con una clase resourcebundle que creemos nosotros mismos para recuperar los valores en nuestras JSPs; o de la manera que lo gestiona OpenCms internamente que es poniendo ${key.nombre_propiedad}, ya explicaremos dónde.

Además de estas dos carpetas es interesante saber que en la carpeta *resources* se ponen las imágenes, si es que necesitamos alguna; en la carpeta *elements* podemos meter elementos comunes, por ejemplo, si lo que queremos tener una JSP común e incluirla en otras página mediante un *include*, por ejemplo la index de la web, la pondremos en dicha carpeta.

Hecho esto, para usar la nueva funcionalidad tan sólo debemos incluir los paquetes en cualquier JSP que hayamos creado, instanciar la clase y llamar a sus métodos.

Pongamos que hemos creado un jar que contiene el paquete, com.opencmshispano.modules.holamundo. Para invocarlo desde una jsp se haría así:

```
<%@page buffer="none" session="false" import="org.opencmshispano.holamundo.*"
%>
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<%
out.println(holaMundo.saludar())) ;
%>
```

4. Crear un nuevo punto de administración

En algunas ocasiones tendremos la necesidad de añadirle, no sólo una funcionalidad sino la administración de la misma, lo que llamamos punto de administración. Por ejemplo, si hemos creado una funcionalidad para mostrar encuestas en la web, necesitaríamos administrarlas: crear las encuestas, borrarlas, activarlas o desactivarlas, etc.

Una vez hayamos creado nuestro nuevo módulo, como hicimos en el paso 3, pasaremos a crear su punto de administración dónde necesitaremos seguir los siguientes pasos.

Dentro de /system/workplace/, que cuelga de la raíz del VFS de OpenCms, existen dos carpetas, una es la *admin* y la otra *administration*; en cada una crearemos una carpeta con el nombre de nuestro módulo.

En la subcarpeta de *admin* será donde crearemos nuestras JSP para la gestión del módulo. A dicha carpeta le pondremos la propiedad *NavText*, esta mostrará el título de nuestro módulo y *NavImage*, que mostrará el icono. Aquí podemos hacer uso de la información del fichero *properties* que creamos anteriormente, donde bien introducimos nosotros los datos o los cogemos del fichero properties poniendo ${key.nombre_de_la_propiedad}. A todas las JSPs se les pueden poner las propiedades:

- **Title:** Titulo del modulo. Aquí es dónde tiraremos del *.properties* y pondremos ${key.titulo}
- **NavText**: Titulo publico del modulo.
- **NavPos:** Posicionamos el icono en la posición que nos interese.
- **NavInfo:** Información adicional del modulo.
- **NavImage:** Url de la imagen que aparecerá como icono.
- **Description:** Descripción del modulo.
- **admintoolhandler-class:** Clase que maneja el modulo. Se puede usar la clase por defecto: org.opencms.workplace.tools.CmsDefaultToolHandler
- **default-file:** Fichero por defecto que se lanzará al entrar en el modulo. Suele ser un estandar usar un index.jsp como fichero por defecto.

Como mínimo tendremos que asignarle un valor a las siguientes propiedades**:** *title, adminitoolhandler-class* y *default-file*.

La manera de cargar las JSP es mediante otra JSP que se llama *admin-main.jsp.* Ésta se encuentra en /system/workplace/views/admin. Hay varias formas de realizar la redirección correctamente a esta JSP. La más sencilla es creando subcarpetas en la carpeta creada dentro de *administration*. Para llevar a cabo la redirección crearemos tantas carpetas como JSPs haya en *admin*, exceptuando la que se haya establecido como inicial.

Dichas carpetas tendrán el mismo nombre que el *title* la JSP a la que esté asociada. Lo cual creará un icono para cada carpeta dentro de nuestro módulo en la vista de administración.

Esto nos conduce a un dilema, si cada carpeta será representada por un icono en la vista de administración, pero por alguna razón quiero que exista tal icono, ¿qué debemos hacer?

Anteriormente hemos mencionado la propiedad *admintoolhandler-class* de las carpetas, lo único que indica es que *handler* o manejador que gestionará el punto de administración. Éste implementa una interfaz cuyos métodos son *isVisible* y *isAdmin*. El que nos interesa implementar es *isVisible*, donde pondremos la lógica que evaluará qué iconos son los que queremos que se muestren en nuestro módulo.

Una posible implementación de un handler sería:

```
package org.opencmshispano.module.holamundo.handler;

import org.opencms.file.CmsObject;
import org.opencms.workplace.tools.A_CmsToolHandler;

public class HolaMundoHandler extends A_CmsToolHandler
{

    public boolean isEnabled(CmsObject cms)
    {
        return true;
    }

    public boolean isAdmin(CmsObject cms)
    {
        return true;
    }


    public boolean isVisible(CmsObject cms)
    {
            boolean visible=true;
            if(!getPath().equals("/holaMundo/adios”))
            {
                    visible=false;
            }
        return visible;
    }
}
```

Los dos primeros métodos no se han implementado en este caso. El *isEnabled* te permite implementar la lógica de cuando se puede “hacer clic” sobre un icono. El método *isAdmin* te permite ver si el usuario es administrador o no y por tanto permitirle o no gestionar el módulo. Finalmente *isVisible* implementa la lógica para ver si el icono es o no visible.

Nota: Para que se vean los cambios en la vista de administración hay que publicar los cambios de las propiedades que afecten al aspecto visual o al

manejador del punto de administración hay que reiniciar el *workplace* tal y como se explico cuando instalamos el módulo de traducción al castellano.

Para terminar nos falta hacer las JSPs del punto de administración, que se encuentran en la subcarpeta de admin, tienen una estructura determinada.

Estructura de las jsps de administración

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %><%@ page
import="
        org.opencms.i18n.*,
        org.opencms.cache.*,
        org.opencms.flex.*,
        org.opencms.main.*,
        org.opencms.workplace.CmsDialog,
        org.opencms.jsp.CmsJspActionElement,
        java.util.*"
%><%
        CmsJspActionElement jsp = new CmsJspActionElement(pageContext,
request, response);
        CmsDialog wp = new CmsDialog(jsp);

        if (wp.getAction()==CmsDialog.ACTION_CANCEL) {
                ////////////////// ACTION: cancel button pressed: redirect to closelink
                wp.actionCloseDialog();
                return;
        }
%>
<%= wp.htmlStart("administration/index.html") %>
<%= wp.bodyStart(null) %>

<%= wp.dialogStart() %>
<%= wp.dialogContentStart(wp.key("key_nuevo_modulo")) %>
<%
                        //Código especifico de nuestra JSP
%>

<%= wp.bodyEnd() %>
<%= wp.htmlEnd() %>
```

Como ya hemos mencionado la forma de redireccionar de una JSP a otra es algo especial. Una forma de hacerlo correctamente es haciendo un *sendRedirect*, pasándole como argumento a este método, la siguiente linea:

admin-main.jsp?path=%2Fcarpeta_en_administration%2Fsubcarpeta.

Es decir, si queremos redireccionar a adios.jsp de nuestro módulo HolaMundo, tendremos que pasarle admin-main.JSP?path=%2Fholamundo%2Fadios.

Nota: En todas aquellas carpetas que queramos que sean accesibles mediante iconos o redireccionamiento, hay que ponerles la propiedad **admintoolhandler-class.**

Clase HolaMundo

```
package org.opencmshispano.holamundo;

Public class HolaMundo
{

   public static String saludar()
  {
    return (“Hola mundo” );
  }

 public static String adios()
 {
  return (“Adios” );
}

}
```

workplace.properties

```
title.holamundo = Hola mundo
icono.holamundo = ../opencms/system/modules/org.opencmshispano.holamundo/resources/images.jpg
```

Index.jsp

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %><%@ page import="
        org.opencms.i18n.*,
        org.opencms.cache.*,
        org.opencms.flex.*,
        org.opencms.main.*,
        org.opencms.workplace.CmsDialog,
        org.opencms.jsp.CmsJspActionElement,
        org.opencmshispano.holamundo.*,
        java.util.*"
%><%
        CmsJspActionElement jsp = new CmsJspActionElement(pageContext, request,
response);
        CmsDialog wp = new CmsDialog(jsp);

        if (wp.getAction()==CmsDialog.ACTION_CANCEL) {
                ////////////////// ACTION: cancel button pressed: redirect to closelink
                wp.actionCloseDialog();
                return;
        }
%>
<%= wp.htmlStart("administration/index.html") %>
<%= wp.bodyStart(null) %>

<%= wp.dialogStart() %>
<%= wp.dialogContentStart(wp.key("title.holamundo")) %>
<%
out.println(HolaMundo.saludar());
%>

<%= wp.bodyEnd() %>
<%= wp.htmlEnd() %>
```

Adios.jsp

```
<%@ taglib prefix="cms" uri="http://www.opencms.org/taglib/cms" %><%@ page import="
        org.opencms.i18n.*,
        org.opencms.cache.*,
        org.opencms.flex.*,
        org.opencms.main.*,
        org.opencms.workplace.CmsDialog,
        org.opencms.jsp.CmsJspActionElement,
        org.opencmshispano.holamundo.*,
        java.util.*"
%><%
        CmsJspActionElement jsp = new CmsJspActionElement(pageContext, request,
response);
        CmsDialog wp = new CmsDialog(jsp);

        if (wp.getAction()==CmsDialog.ACTION_CANCEL) {
                ////////////////// ACTION: cancel button pressed: redirect to closelink
                wp.actionCloseDialog();
                return;
        }
%>
<%= wp.htmlStart("administration/index.html") %>
<%= wp.bodyStart(null) %>

<%= wp.dialogStart() %>
<%= wp.dialogContentStart(wp.key("title.holamundo")) %>
<%
out.println(HolaMundo.adios());
%>

<%= wp.bodyEnd() %>
<%= wp.htmlEnd() %>
```