|  |  |
| --- | --- |
|  |  |

**Curso Taller Liferay Portal HCUVA**

*Oct / Nov 2014*

Contenido

[1 Horario 5](#_Toc404895413)

[2 Estructura del curso 6](#_Toc404895414)

[3 Introducción a Liferay 8](#_Toc404895415)

[3.1 Qué nos ofrece Liferay? 8](#_Toc404895416)

[3.2 Ayuda al desarrollador 8](#_Toc404895417)

[3.3 Versiones Liferay Portal CE 9](#_Toc404895418)

[4 Primera visualización de Liferay Portal 10](#_Toc404895419)

[5 Gestión Visual Liferay Portal 11](#_Toc404895420)

[5.1 Panel de Control 11](#_Toc404895421)

[5.2 Administración de un Sitio Web 13](#_Toc404895422)

[5.2.1 Creación de un Sitio Web 13](#_Toc404895423)

[5.2.2 Añadir Páginas 14](#_Toc404895424)

[5.2.3 Añadir Portlets 15](#_Toc404895425)

[5.2.4 Definir permisos a un portlet 15](#_Toc404895426)

[5.2.5 Incluir contenidos web 16](#_Toc404895427)

[5.2.6 Añadir Imágenes y Documentos 17](#_Toc404895428)

[5.2.7 Etiquetar y Categorizar contenidos. 18](#_Toc404895429)

[5.2.8 Búsqueda de contenidos 18](#_Toc404895430)

[5.2.9 Publicador de Contenidos 19](#_Toc404895431)

[5.2.10 Cambiar layout 20](#_Toc404895432)

[5.2.11 Cambiar tema de apariencia 21](#_Toc404895433)

[5.2.12 Crear nuevos usuarios 22](#_Toc404895434)

[5.2.13 Asignar permisos a un usuario. 23](#_Toc404895435)

[5.2.14 Crear nuevos roles 24](#_Toc404895436)

[5.2.15 Plantillas 26](#_Toc404895437)

[5.2.16 Estructuras 28](#_Toc404895438)

[5.2.17 Listas Dinámicas de Datos 32](#_Toc404895439)

[5.3 Workflow 36](#_Toc404895440)

[6 Entorno Desarrollo, Configuración Eclipse 38](#_Toc404895441)

[6.1 Instalación Liferay IDE y Liferay SDK 39](#_Toc404895442)

[6.2 Configuración servidor 39](#_Toc404895443)

[6.3 Primera puesta en marcha 40](#_Toc404895444)

[7 Debug código fuente 40](#_Toc404895445)

[8 Cambio a base de datos MySql 40](#_Toc404895446)

[9 Tipos de Plugins 41](#_Toc404895447)

[10 Desarrollo de Plugins con Liferay IDE 42](#_Toc404895448)

[10.1 Primer ejemplo: Hola Formación 42](#_Toc404895449)

[10.2 Creación de un Theme 44](#_Toc404895450)

[10.3 Creación de un Layout 48](#_Toc404895451)

[10.4 Creación Hook 49](#_Toc404895452)

[10.4.1 Modificar JSP Original 49](#_Toc404895453)

[10.4.2 Añadir texto a un jsp 52](#_Toc404895454)

[10.4.3 Modificar JSPs para diferentes sitios 52](#_Toc404895455)

[10.4.4 Modificar Propiedades del Portal 56](#_Toc404895456)

[10.4.5 Incluir Acciones a Eventos del Portal 57](#_Toc404895457)

[10.4.6 Extender los Servicios de Liferay 59](#_Toc404895458)

[10.4.7 Modificación de ficheros de traducción 61](#_Toc404895459)

[10.5 Creación de Portlets 62](#_Toc404895460)

[10.5.1 Portlet con dos modos 62](#_Toc404895461)

[10.5.2 IPC Portlet 65](#_Toc404895462)

[10.5.3 Ejemplos Portlet PHP 72](#_Toc404895463)

[10.5.4 Ejemplo Portlet JSF Bridge 76](#_Toc404895464)

[10.6 Service Builder Portlet 80](#_Toc404895465)

[10.6.1 Definir nuestro ORM y configurar Service.xml 80](#_Toc404895466)

[10.6.2 Servicios Locales 84](#_Toc404895467)

[10.6.3 Llamando a nuestros servicios locales 93](#_Toc404895468)

[10.6.4 Interfaz 99](#_Toc404895469)

[10.6.5 Ampliando funcionalidad 105](#_Toc404895470)

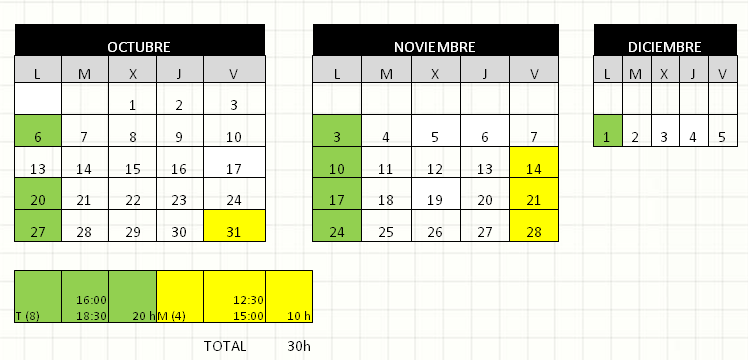
[10.6.6 Acceso a recursos 117](#_Toc404895471)

[10.6.7 Scope 124](#_Toc404895472)

[10.6.8 Servicios Remotos 126](#_Toc404895473)

[10.6.9 Custom SQL Queries 131](#_Toc404895474)

# Horario



# Estructura del curso

Bloque 1 ( 25% )

- Introducción a Liferay

- ¿Qué nos ofrece Liferay?

- Ayuda al desarrollador

- Versiones Liferay Portal

- Primera visualización de un Portal

- Gestión de contenidos

- Niveles de gestión ( usuario, sitio y portal )

- Sitios, Organizaciones, Grupos de trabajo …

- Roles

- Páginas

- Documentos, tipos

- Estructuras

- Plantillas

- Etiquetas

- Categorías

- Flujos de Publicación

Bloque 2 ( 25% )

- Flujos de trabajo

- diseño

- formularios

- listas dinámicas

- notificaciones

- gestión

- roles

Bloque 3 ( 50% )

- Programación de Plugins

- Entorno de Desarrollo, Configuración de Eclipse

\* Instalación Liferay IDE y Liferay SDK

\* Configuración Servidor

\* Primera puesta en marcha

- Tipos de Plugins

- Primer Ejemplo : Hola Mundo

- Portlet con dos modos

- IPC Portlet

- Portlet con JSF Bridge

- Service Builder Portlet

- Otro conocimiento necesario

- Debug código fuente

- Configuración base de datos

- Creación de un Theme

\* Modificación de estructura

\* Modificación de estilos

- Creación de un Layout

- Creación de Hooks

\* Modificación de propiedades

\* Modificación de JSPs

\* Ampliación de acciones

\* Modificación idiomas

# Introducción a Liferay

## Qué nos ofrece Liferay?

Se basa en su página web: [www.liferay.com](http://www.liferay.com)

Productos:

* + Liferay Portal
  + Liferay Social Office
  + Liferay Sync

Otros Proyectos:

* Alloy UI
* Liferay FACES
* Liferay IDE
* Integración OSGi, Sandboxing, Cloud Services ….

## Ayuda al desarrollador

Liferay tiene diferentes páginas donde los desarrolladores pueden encontrar información útil:

* + Documentación. Para la versión 6.1 está en <http://www.liferay.com/es/documentation/liferay-portal/6.1/user-guide>
    - Guía de Usuario
    - Desarrollo
  + Comunidad, Foros, Blogs. Los Foros de Liferay son una muy buena forma de encontrar información específica a nuestras dudas ya que existe una comunidad bastante activa: <http://www.liferay.com/es/community/forums>

A través de los blogs

<http://www.liferay.com/es/community/blogs> también se tiene información actualizada y de desarrollo tanto del personal de Liferay como de usuarios.

* + Marketplace. <http://www.liferay.com/es/marketplace> Centro común de portlets oficiales o de la comunidad tanto gratuitos como de pago.

## Versiones Liferay Portal CE

Liferay se nos ofrece principalmente de dos formas distintas:

* + Versión Bundled: Todo en uno, incluye el servidor de aplicaciones elegido más el código del Portal con algunos portlets preinstalados. La versión 6.1 fue la última en incluir en el paquete una versión Java por lo que es recomendable tenerla instalada con anterioridad en nuestro sistema operativo.
  + Podemos también instalar Liferay como una aplicación independiente contenida en un .war en un Servidor de Aplicaciones que ya tengamos ( aunque no es recomendable que Liferay lo comparta con otras aplicaciones ). A nuestra disposición también disponemos de los scripts de creación de la base de datos de forma independiente, código fuente, javadoc …

# Primera visualización de Liferay Portal

Para empezar ver por primera vez cómo es Liferay Portal, descargamos de

<http://www.liferay.com/es/downloads/liferay-portal/available-releases>

la versión 6.1.2 CE GA3 bundled. También está disponible en la carpeta “Formacion” disponible por el profesor durante el curso. Los pasos a seguir son

* + Obtener la versión 6.1.2 CE GA3 bundled con Tomcat
  + Descomprimir y acceder a la carpeta %TOMCAT\_HOME%/bin
  + Tener preconfigurada la variable de entorno JAVA\_HOME o JRE\_HOME ( en la máquina virtual del curso no está preinstalada )
  + Lanzar “startup.sh”. Esperar y se nos abriré la página de bienvenida cuando termine el despliegue.

Nota 1: Observar las fases del despliegue ( Descriptor, carga propiedades, detección base de datos, creación de tablas si fuera necesario, carga de plugins ) . En caso de no existir las tablas en la base de datos, las crea de forma inicial.

Nota 2: Por defecto la versión Bundled está preconfigurado para utilizar 1 Gb de memoria. Asegurarse que al menos la máquina tiene 2 megas para poder desplegar con garantías.

La primera vez que se lanza encontramos un asistente de configuración básico, que salvo cambiar el idioma y quitar la creación de datos de muestra no tocaremos nada más.

A continuación configuraremos la cuenta del administrador máximo del portal, que será [test@liferay.com](mailto:test@liferay.com) y de contraseña pondremos “test”. Aceptar términos de uso y pregunta de seguridad si fuera necesario.

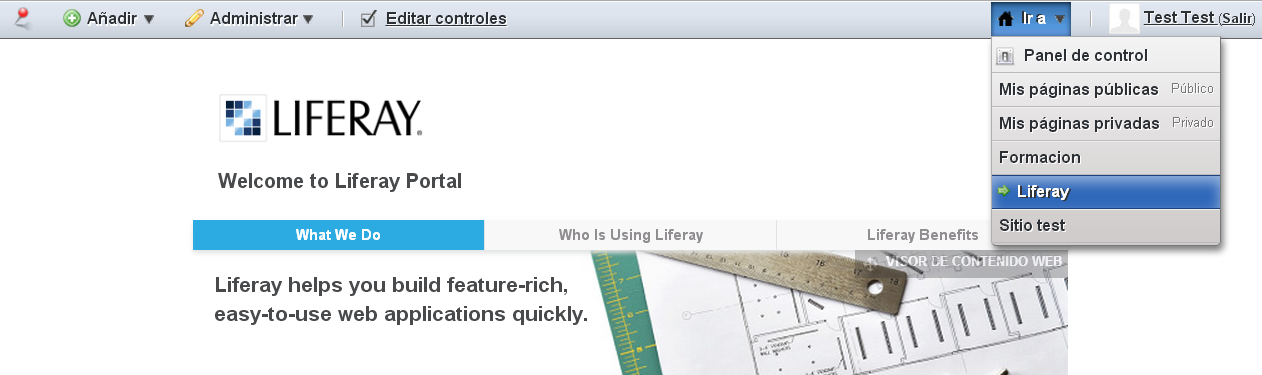
No entraremos en detallar mucho del Usuario final sobre cómo usar Liferay sólo lo básico ya que existe un curso específico para ello. Para desarrollo es mucho más interesante ver la estructura de carpetas que tenemos ahora:

* /Webapps , donde se encuentra el código de Liferay y demás plugins
* /ROOT, código de Liferay. Carpetas importantes como /html/portlet, /html/theme, /WEB-INF/classes, /WEB-INF/lib
* /data
  + Hsql, base de datos Hypersonic
  + Jackrabbit, motor de almacenamiento
  + Lucene, motor de indexación
  + ….

# Gestión Visual Liferay Portal

## Panel de Control

Al panel de control de cualquier usuario que accede a la plataforma se accede a través del dockbar de la parte superior:



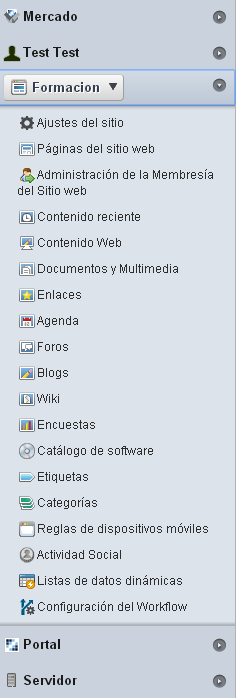
, una vez dentro podremos distinguir 4 niveles de gestión de nuestro portal (algunos pueden no verse si no tenemos permisos para ello).



En la gestión de usuario podremos comprobar y editar todo lo referente a los datos de usuario así como las páginas privadas y públicas en el caso que las tuviéramos.

Además, si el workflow estuviera activado tendríamos todo lo referente a nuestras solicitudes o las tareas que tenemos pendientes.

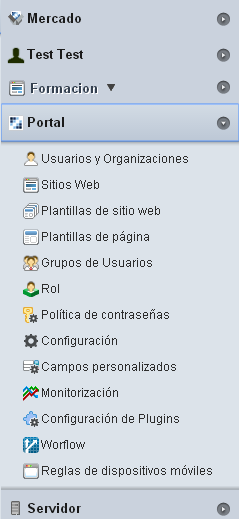
Si pertenecemos a algún sitio u organización, a continuación podremos gestionarlos con la siguiente pestaña.



Según los permisos que tengamos podremos administrar más o menos aplicaciones. Las más importantes y a destacar serían:

* Páginas del sitio web: Gestionamos nuestras páginas con su apariencia, distribución …
* Contenido Web: Administramos todos los contenidos introducidos. Además aquí se gestionan las estructuras que mostraremos más adelante del curso
* Documentos y Multimedia: Controlamos todos los documentos e imágenes subidas al portal así como su organización.
* Etiquetas y Categorías: Para indicar las mismas y así poder tener una mayor organización de nuestro contenido

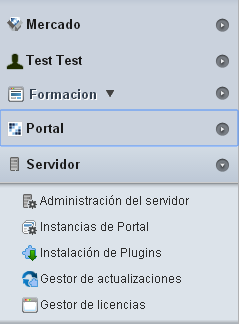
El siguiente nivel que nos ofrece Liferay es para gestionar el Portal, por tanto todo lo que se defina a éste nivel afectará a toda nuestra instalación.



Aquí es donde gestionaríamos los usuarios de nuestro portal, así como sus permisos,pertenencia … También donde podremos crear nuevos sitios web u organizaciones.

Podremos crear plantillas tanto a nivel de sitio como de página e incluso podremos definir los permisos de nuestros roles o añadir algunos nuevos.

La pestaña de configuración nos será muy útil por ejemplo para definir el idioma principal de nuestro Portal o los parámetros de autentificación.



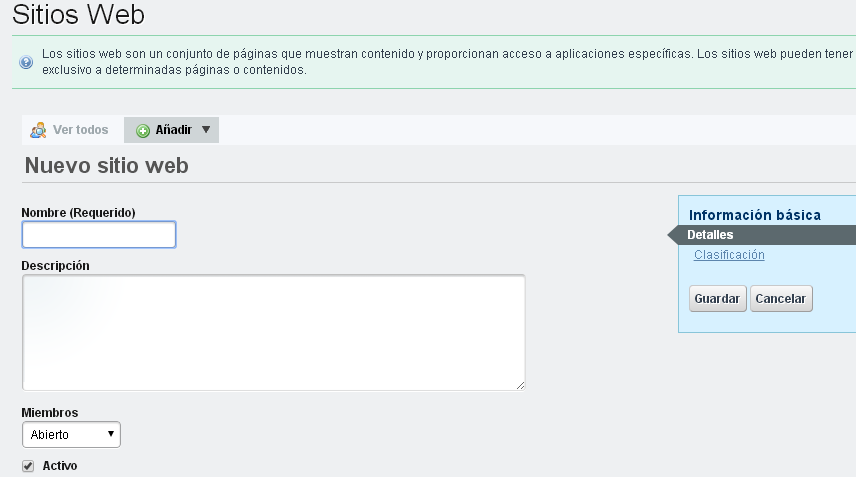
Por último tendremos más información sobre algunas configuraciones de nuestro servidor: Memoria ocupada, memoria disponible, variables de configuración de la máquina, de Liferay …

A éste nivel es donde debemos configurar un servidor de correo para el envío general de la instancia o establecer una conexión con OpenOffice para la conversión de ficheros, por ejemplo.

## Administración de un Sitio Web

### Creación de un Sitio Web

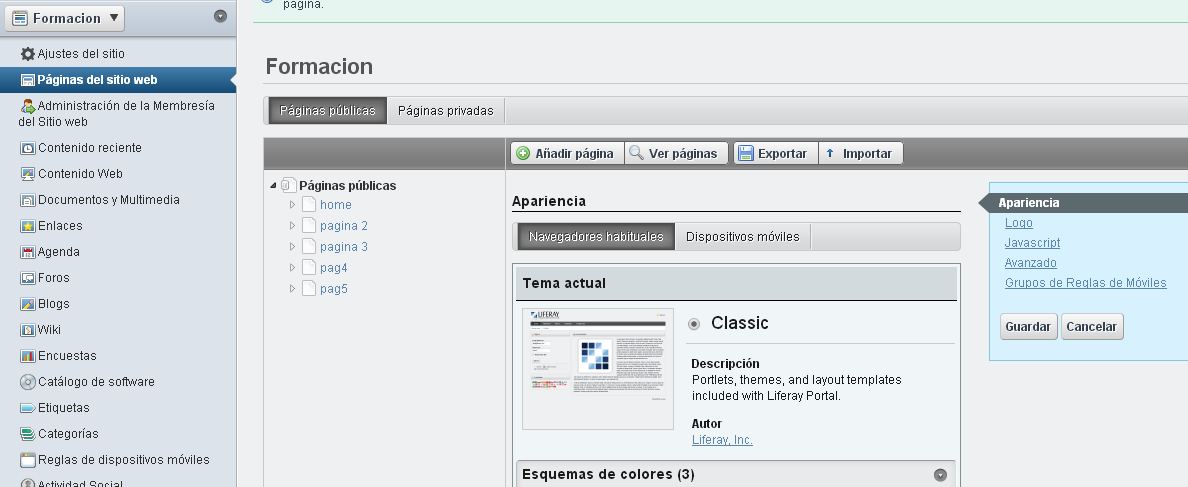
Para crear un sitio web, sólo tendremos que irnos a “Portal” => “Sitios Web” => “Añadir” => “Sitio Web en blanco”



### Añadir Páginas

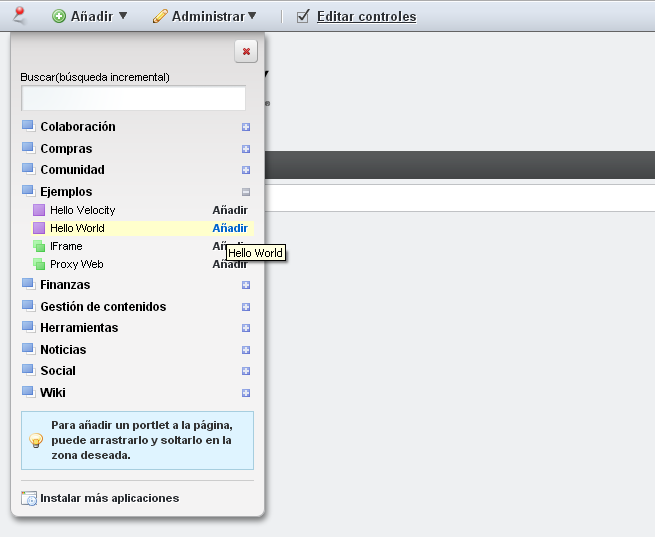
Para añadir la primera página de nuestro Sitio o posteriores, tendremos que señalar en el menú de la izquierda nuestro sitio creado (en el curso se llama “Formación”) y seleccionar “Páginas de sitio web”.

Veremos que tendremos para añadir páginas tanto en el ámbito de páginas públicas como privadas. Es pulsar el botón añadir y ya podremos ir creando nuestro árbol de páginas.



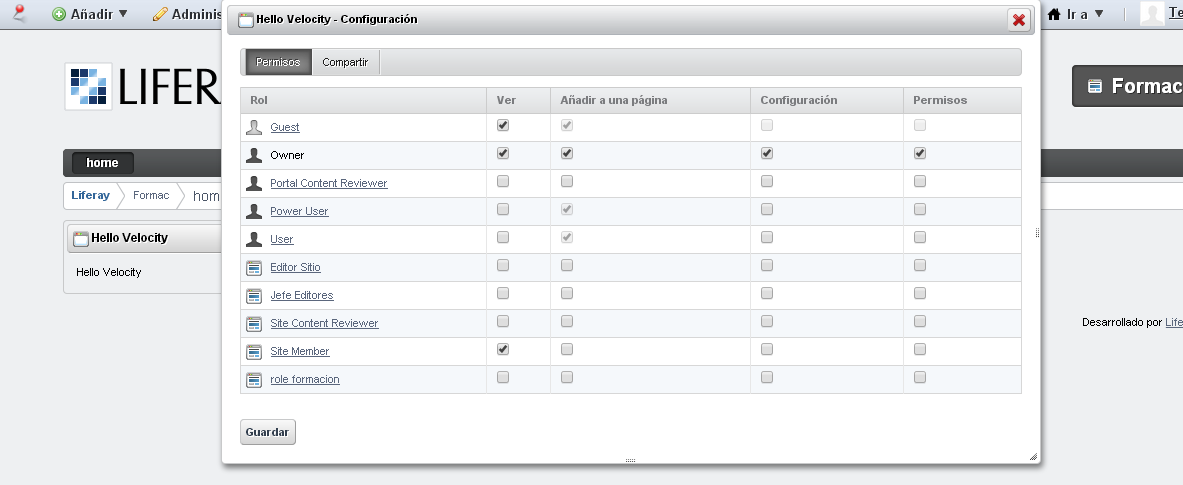
### Añadir Portlets

Si queremos añadir un portlet a una página, sólo nos situaremos en dicha página y, siempre que los permisos lo permitan, seleccionamos en el dockbar “Añadir” => “Más…”



### Definir permisos a un portlet

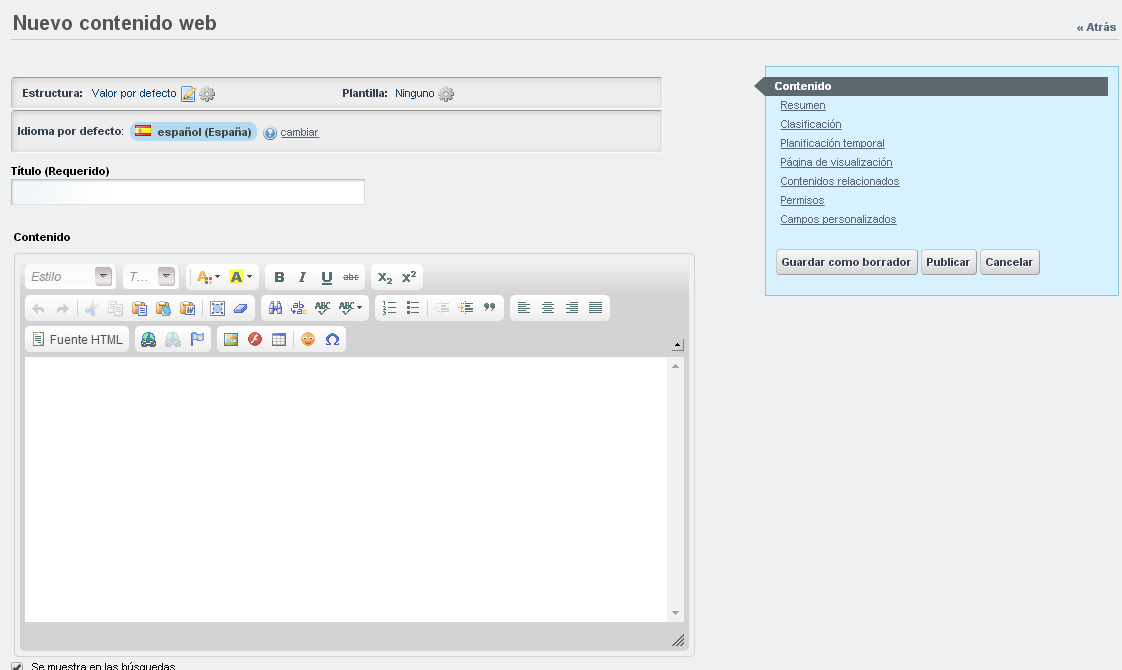
Si queremos por ejemplo ocultar el portlet añadido a los usuarios no logueados deberemos señalar en el icono que parece una herramientas => “Configuración”:



, debemos desmarcar la opción de “Ver” para el rol “Guest” (no logueado).

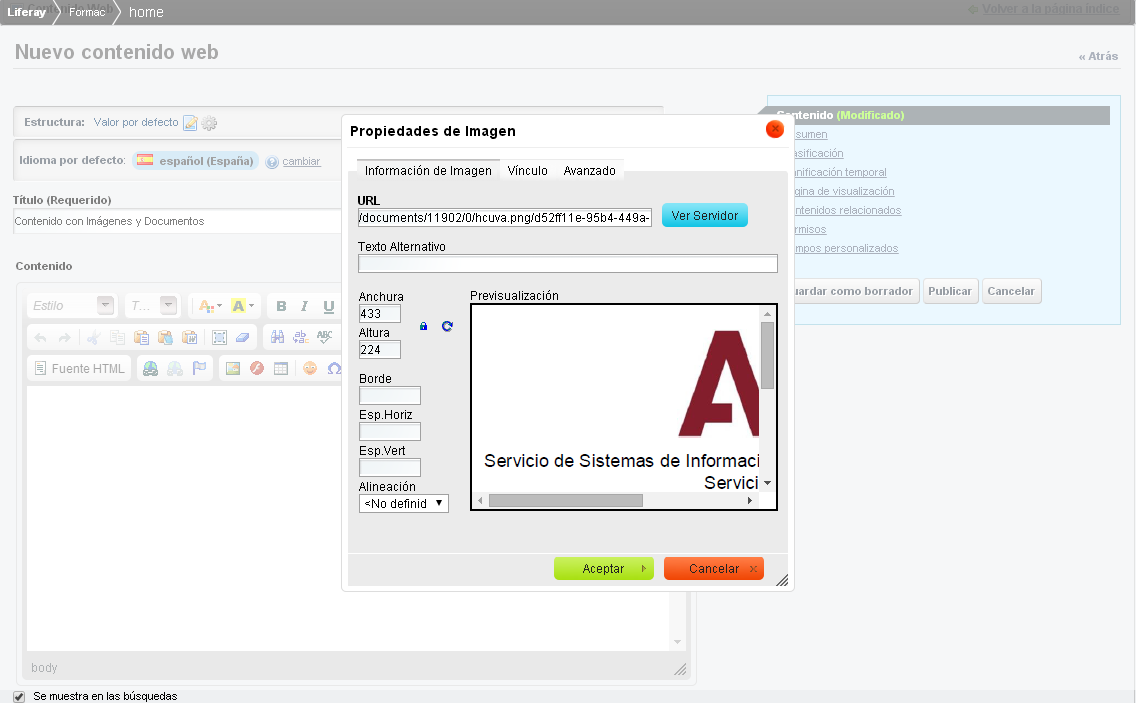
### Incluir contenidos web

La función más usada en un portal es añadir un “Contenido Web”. Seleccionaremos “Añadir” => “Visor de contenido web”. Una vez incluido pulsaremos en “añadir contenido web” y nos mostrará un formulario básico en el que tendremos principalmente un WYSIWYG para incluir nuestro contenido:

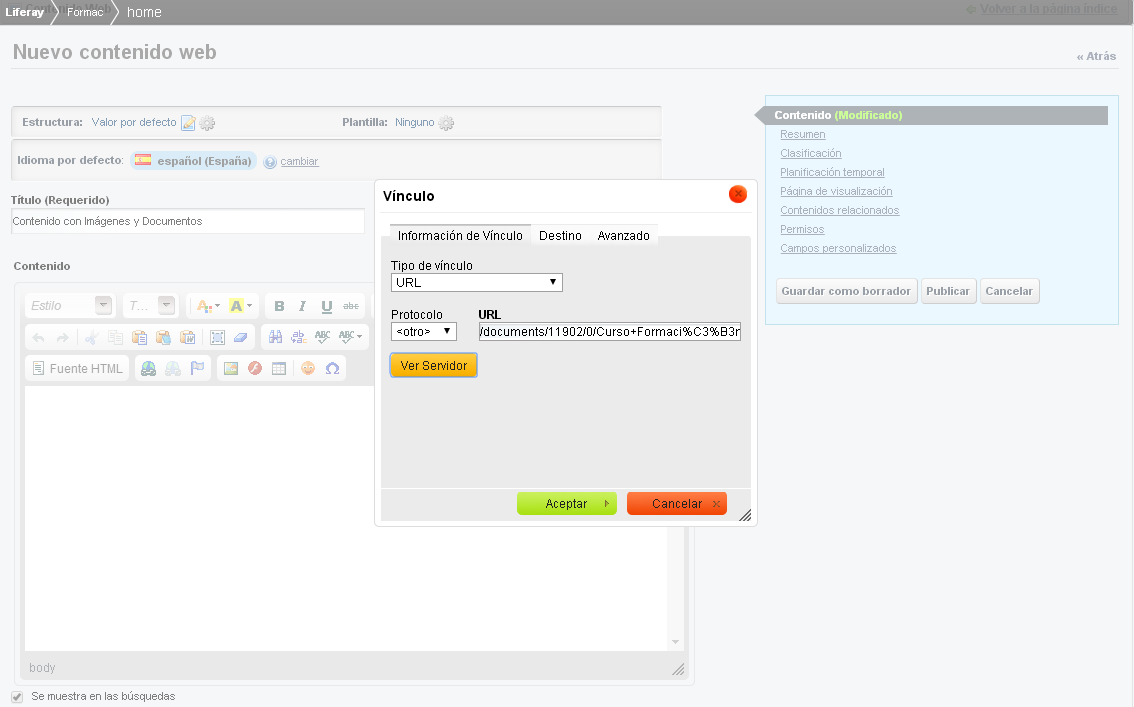


### Añadir Imágenes y Documentos

Si queremos añadir un contenido con imágenes y documentos haremos lo siguiente. En el WYSIWYG seleccionaremos sobre el icono “Imagen”. En la URL de la imagen seleccionaremos “Ver Servidor”. Tendremos una visión del Servidor al cual podemos subir imágenes, crear carpetas… Una vez realizado, seleccionamos una imagen y podemos cambiar las propiedades para mostrarlo.

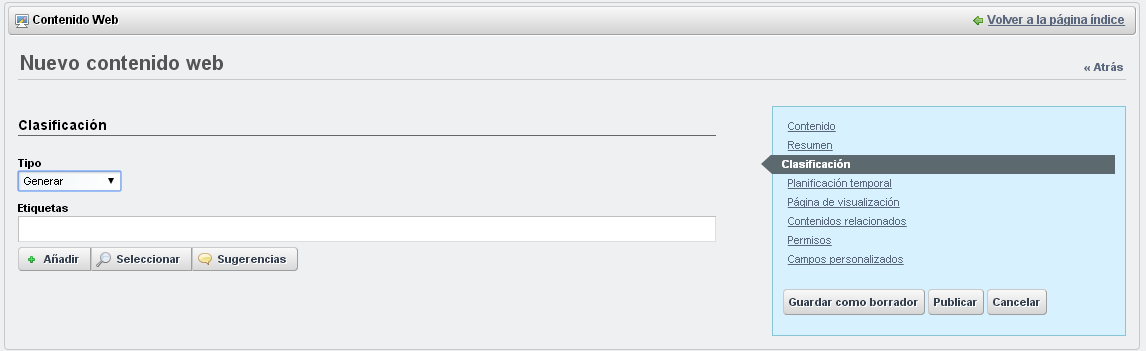


,de una forma similar podemos hacer para imágenes. El icono para el WYSIWYG es el de “Editar/Insertar Vínculo” y pulsamos “Ver Servidor” para añadir nuestros documentos y seleccionarlos.



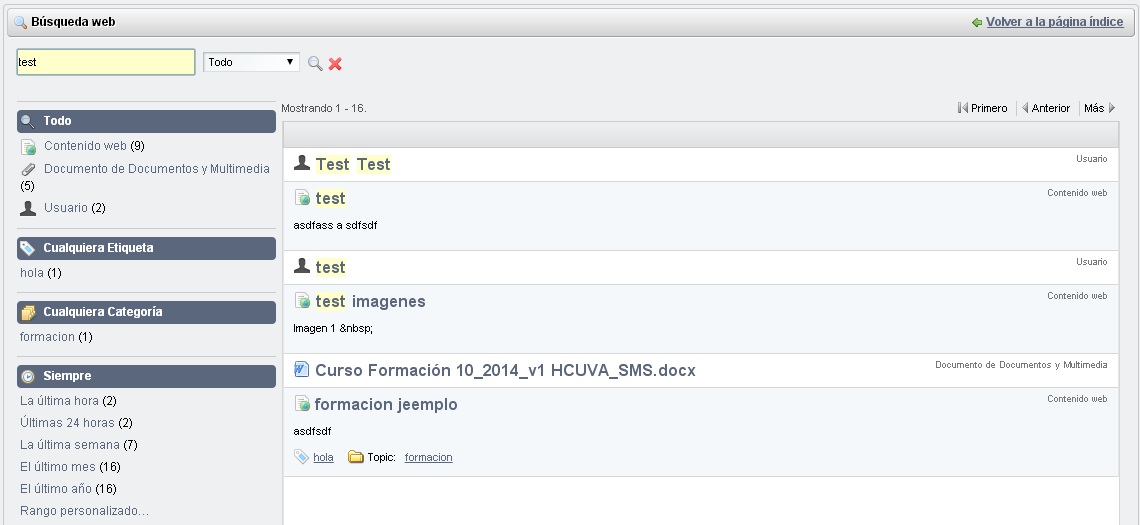
### Etiquetar y Categorizar contenidos.

Una parte importante a la hora de tener un portal con contenido ordenado es tener una estructura sencilla y útil de etiquetas y categorías. Se pueden administrar tanto desde el panel de control (para cada sitio o de forma global) o a la hora de añadir un nuevo contenido en la opción del menú derecho “clasificación”:



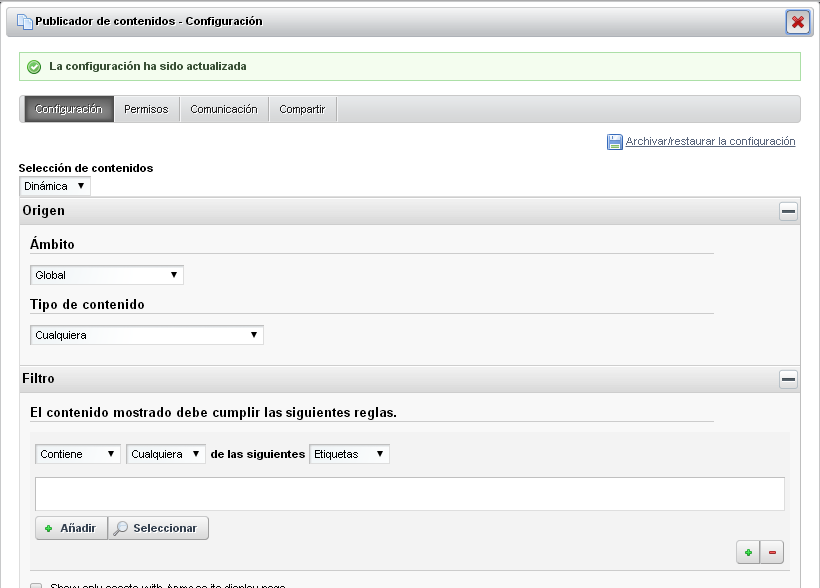
### Búsqueda de contenidos

Cuando tenemos un contenido organizado por etiquetas y categorías, se nos hace mucho más fácil buscarlo y mostrarlo, así como mostrar listados de ellos. Para añadir el buscador, en el dockbar “Añadir” => “Más” => “Búsqueda Web”. Tendremos un buscador dinámico según las opciones señaladas en las opciones izquierdas.



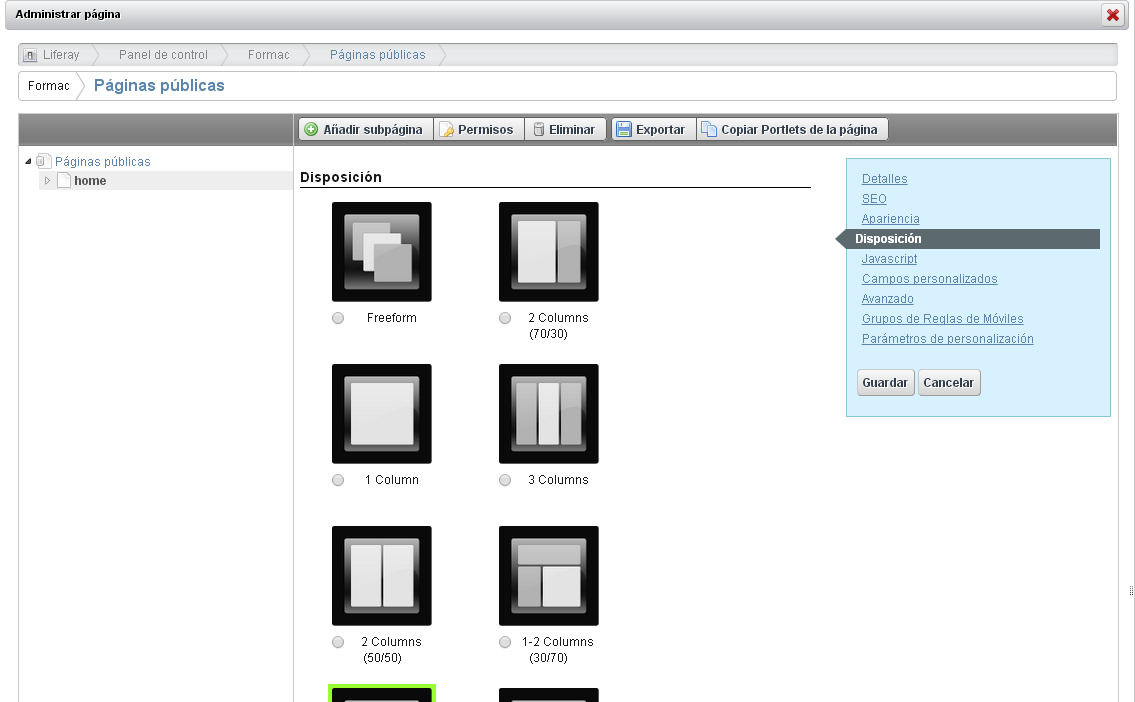
### Publicador de Contenidos

Una de las herramientas más potentes y útiles en Liferay es el Asset Publisher o Publicador de Contenidos. Con él tenemos infinidad de opciones para mostrar a modo de listado el contenido de nuestro portal. Es una herramienta muy versátil de tal forma que podemos mostrarlos por ámbito, fechas, categorías, etiquetas, tipo de contenidos …



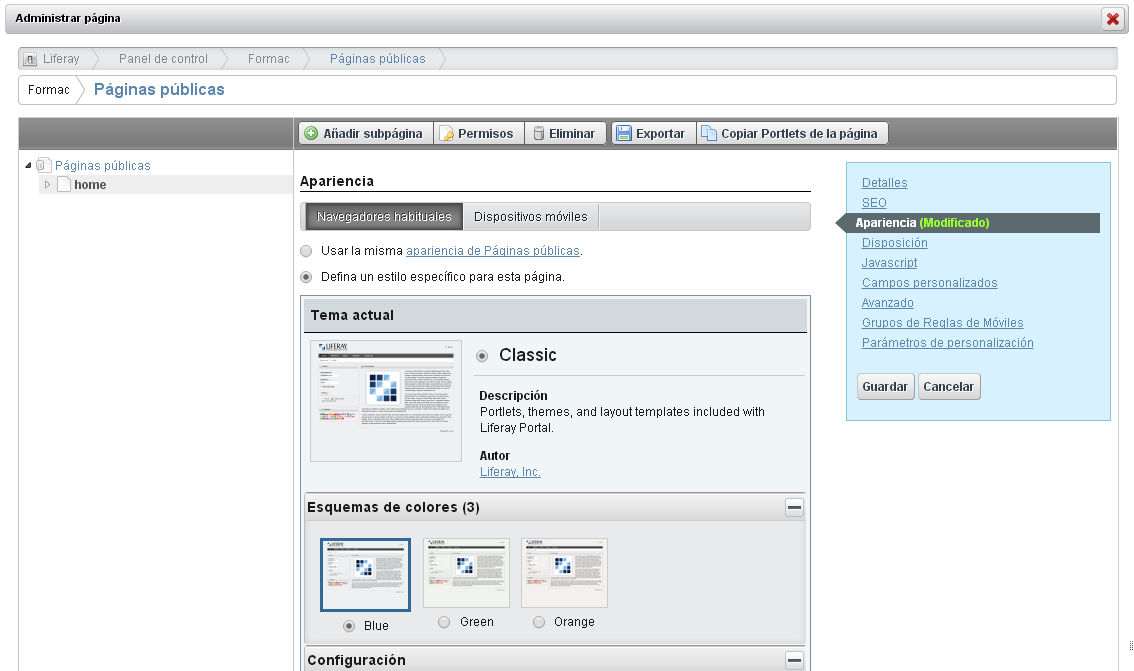
### Cambiar layout

Lo normal cuando vamos a gestionar contenido en nuestras páginas es cambiar la distribución de nuestros “portlets”. Para ello en el dockbar tenemos la opción de “Administrar” => “Disposición de página” y elegir de los disponibles el más indicado:



### Cambiar tema de apariencia

Es habitual cuando administramos nuestro sitio web que una de las primeras cosas que se cambia sea la apariencia del mismo. Para ello sólo tendremos que señalar en el dockbar “Administrar” => “Página” y en el menú derecho “Apariencia”. Ahí podremos señalar cualquiera de los que tenemos disponibles.



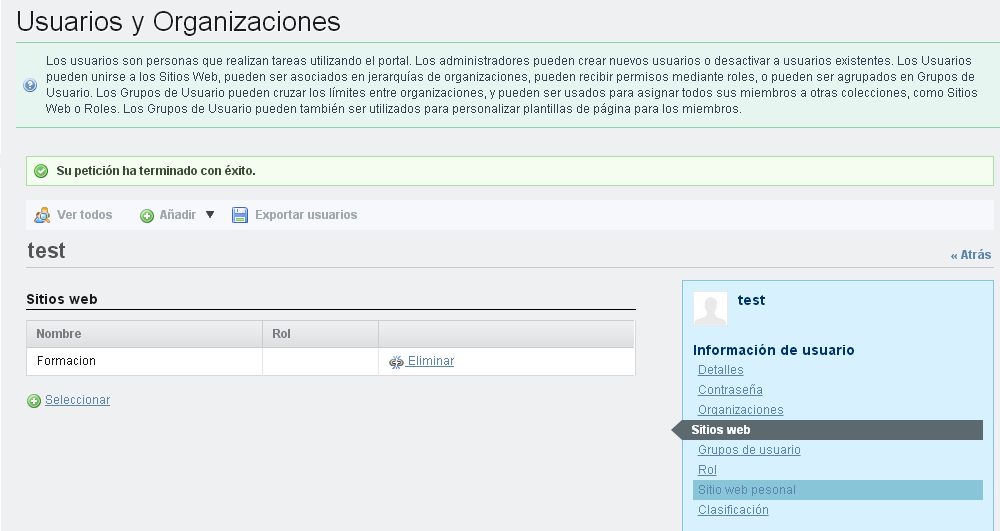
### Crear nuevos usuarios

Para crear nuevos usuarios en nuestro portal nos apoyaremos en la opción respectiva del panel de control. Para ello nos vamos al panel de control (se explicó con anterioridad en éste manual) y en la vista del Portal, señalamos “Usuarios y Organizaciones”. Simplemente pulsando “Añadir” => “Usuario” podremos crear un usuario con los datos que querramos:

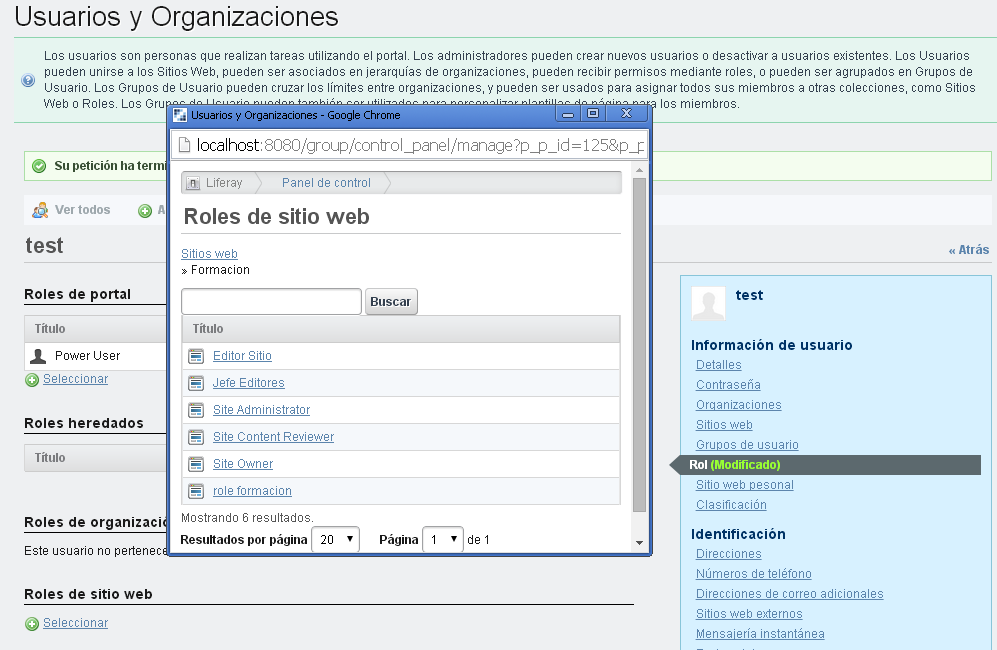


### Asignar permisos a un usuario.

Si al usuario creado queremos añadirle permisos, tenemos que realizar lo siguiente. Una vez creado o marcando “Editar” de uno existente, tendremos que señalar primero que pertenece al Sitio que queremos. El indicar la pertenencia hace que ese usuario automáticamente tenga el permiso de “Miembro de Sitio”.

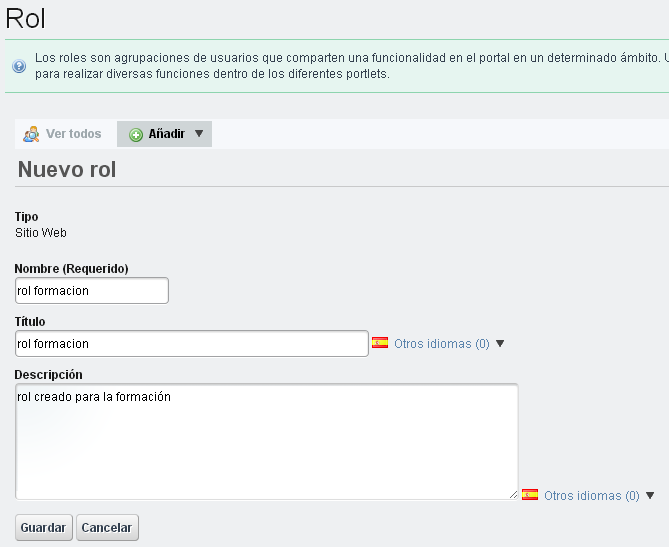
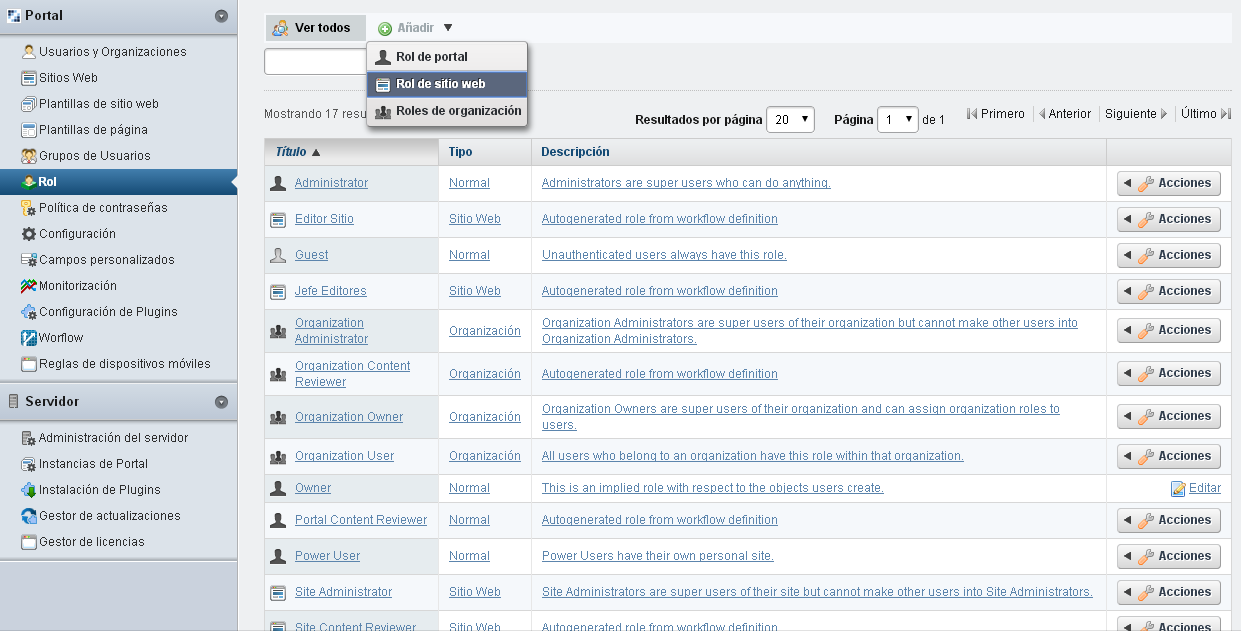


Una vez guardado, en la opción “Rol” podemos asignarle permisos tanto de portal, como de organizaciones o de sitios.

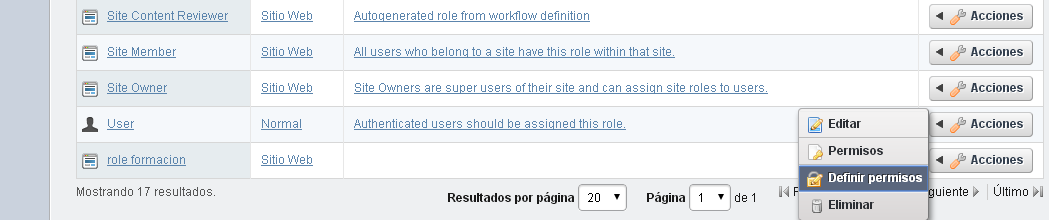


### Crear nuevos roles

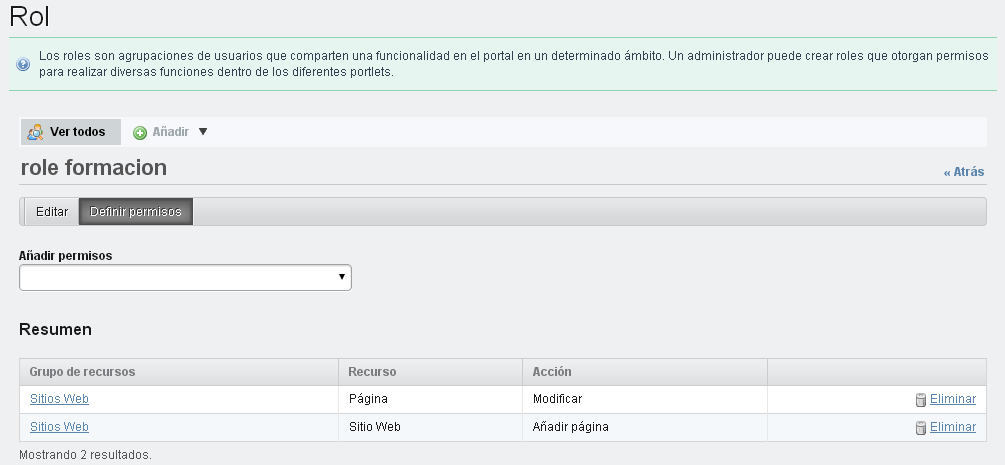
A veces los roles predefinidos que vienen con Liferay no son suficientes para nuestras necesidades. Para ello es sencillo poder crear permisos según el ámbito que queramos: Portal, Organización o Sitio. Para ello en el Panel de Control, nivel Portal señalaremos “Rol” => “Añadir” => “Rol de (Ámbito que queramos)”



Imaginemos que queremos definirle a ese nuevo rol un único permiso que permita añadir páginas al sitio web, pero sólo añadirlas. Por tanto deberemos ir a roles y sobre el que queremos indicar “Definir permisos”:



En la ventana de añadir permisos indicaremos en el desplegable: “Administración de Sitios Web” => “Sitio Web” => “Añadir página” y veremos en un listado los permisos definidos.



### Plantillas

Muchas veces para agilizar la creación de páginas que tienen contenidos similares es conveniente el uso de plantillas. Para ello podremos definir tanto plantillas de página como de sitio. La idea es la misma en ambos casos.

1. Plantillas de página

En el panel de control, nivel de portal => “Plantillas de página” => “Añadir”.



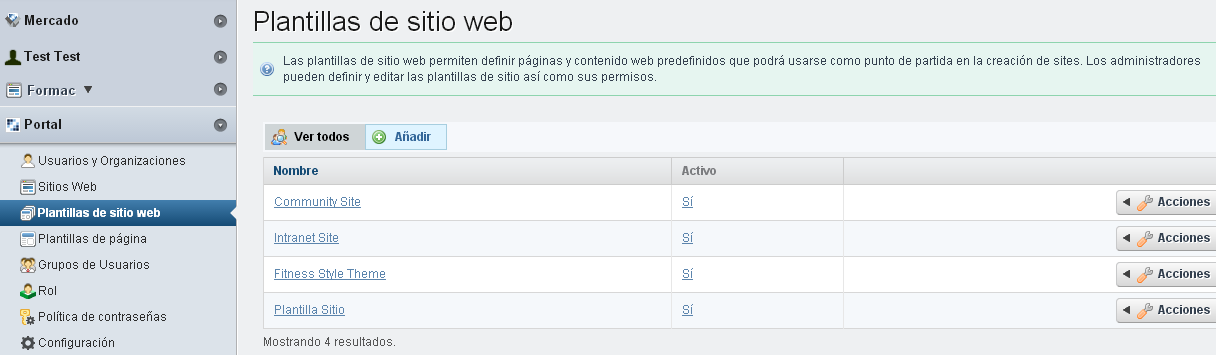
Una vez indicado el nombre, en configuración señalamos “Abrir la plantilla de página”. En ella nos sale una página normal de Liferay en la cual podremos cambiar todo lo que queramos de ella: Contenido, distribución, tema de apariencia… tal cual hemos visto en los puntos anteriores de éste curso.



Una vez tenemos nuestra plantilla, a la hora de crear una nueva página ya podemos seleccionarla y así tendremos toda la página ya preconfigurada.

1. Plantillas de sitio

Al igual que para las páginas, señalamos Panel de Control, vista de Portal => “Plantillas de Sitio Web”. Creamos una nueva plantilla en la que ésta vez podremos realizar la plantilla de una página, sino del sitio web entero. Una vez creado ya podremos añadirlo como plantilla a la hora de crear un sitio nuevo (o editar uno existente) de tal forma que ya tendremos preconfigurado mucho trabajo.



Decir que hay que tener cuidado cuando las usamos y marcamos la opción de “propagar cambios” ya que si la plantilla sufre modificación, todos los sitios/páginas que tengan asignadas esas plantillas verán realizados los cambios automáticamente lo cual a veces no es deseable.

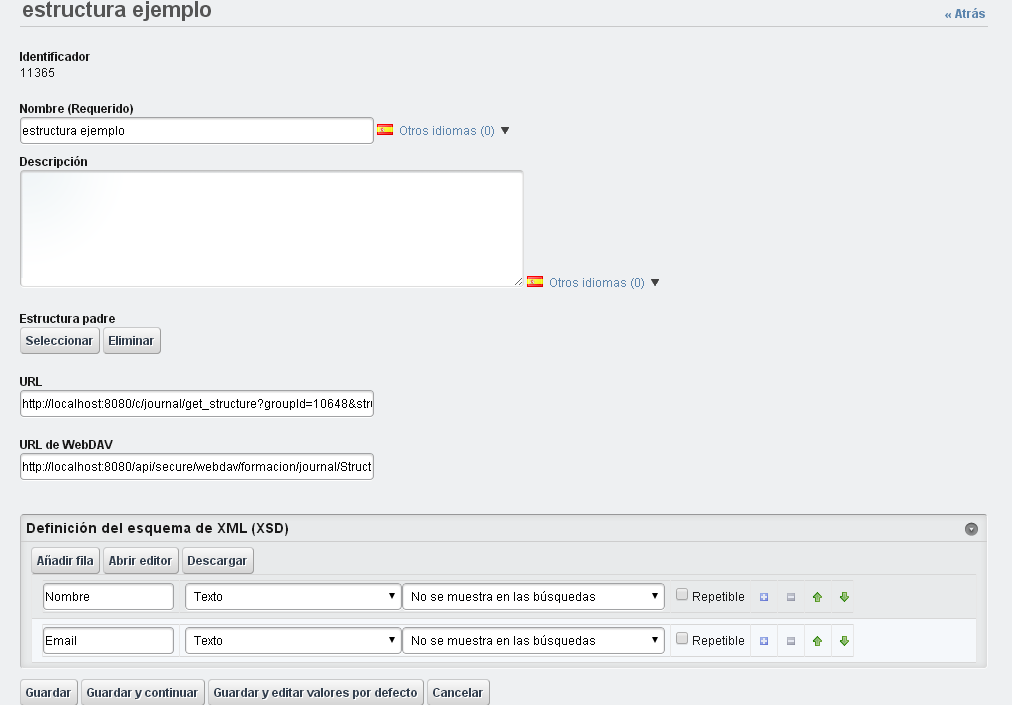
### Estructuras

A veces cuando planteamos una funcionalidad (tipo carrousel de imágenes) lo primero que pensamos es que tendremos que realizar un nuevo desarrollo. Eso hubiera sido así hasta que nos dieron la opción de crear estructuras en Liferay. Es una funcionalidad muy potente y muy útil en el día a día. En general éste contenido nos servirá para que al introducir información, ésta esté guiada y tenga sólo la información que queremos.

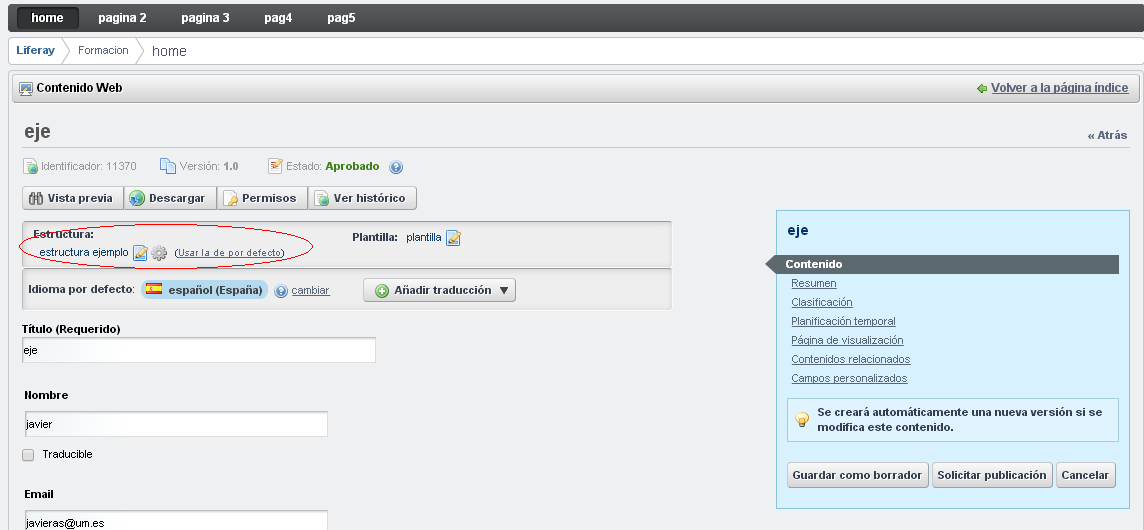
Para crear una nueva estructura, sólo tendremos que irnos al panel de control => nivel de sitio => “Contenido Web”



Al pulsar sobre “Añadir estructura” creamos nuestra definición de los campos que queremos tener. Por ejemplo si quisiéramos decir que cuando un usuario ponga contenido sólo meta “nombre” y su “email”:

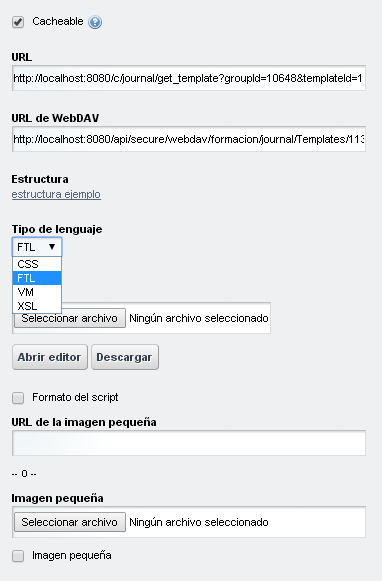


Para comprobar que su funcionalidad es correcta, en la página que queramos añadimos un nuevo contenido web como ya se explicó con anterioridad. En la parte superior podemos señalar nuestra estructura:



Y así podremos introducir contenido con sólo esa información. Si comprobamos tras introducir se nos muestra en el visor de contenido la información sin el formato que buscamos. Para ello las estructuras van íntimamente ligadas a las plantillas de estructuras. Si vemos la imagen anterior, en la parte superior al lado de donde seleccionamos la estructura, podemos seleccionar la plantilla. Le damos a añadir y editamos una nueva.

Son varios los lenguajes que soporta nativamente Liferay para generar plantillas. Aquí realizaremos dos plantillas de ejemplo en dos lenguajes distintos : Velocity y Freemarker. Seleccionamos el que queremos y abrimos el editor.



1. Velocity (VM)

$Nombre.getName()

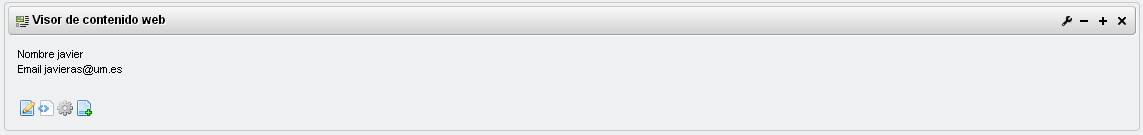
$Nombre.getData()

<br/>

$Email.getName()

$Email.getData()

Y obtenemos:



1. FreeMarker (FTL)

<#assign urlmax = request["render-url-maximized"]>

<#assign namespace = request["portlet-namespace"]>

${Nombre.getName()}

${Nombre.getData()}

<br/>

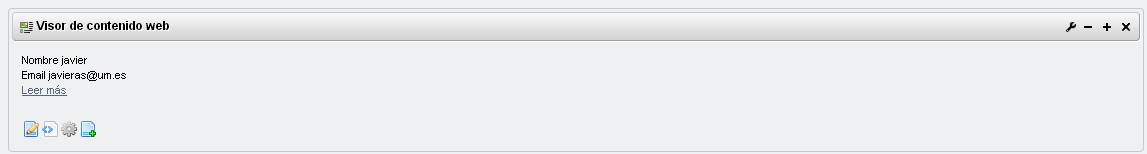
${Email.getName()}

${Email.getData()}

<br/>

<a href="${urlmax}&${namespace}">Leer más</a>

Y se nos mostraría:



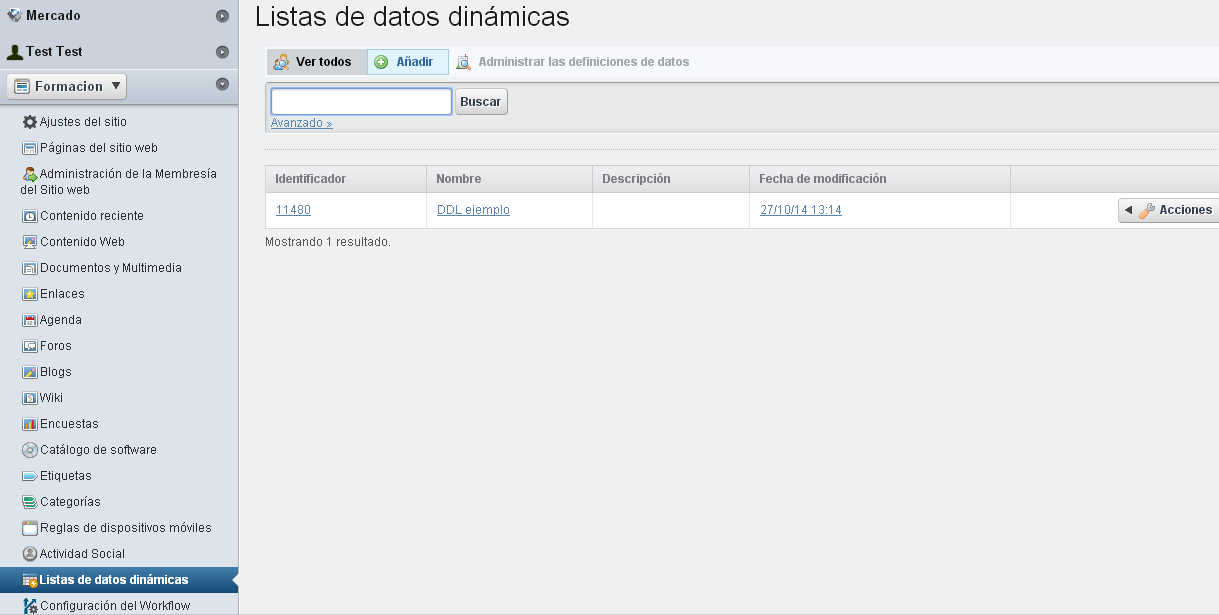
Se pueden obtener variables de diversos ámbitos de la propia plataforma, como en el último ejemplo que hemos recogido variables de la petición de request.

Cómo enlaces de interés para ver qué parámetros y valores podemos coger tenemos:

* + <http://cdn.docs.liferay.com/portal/6.1/javadocs/com/liferay/portal/util/PortalUtil.html>
  + <https://www.liferay.com/es/documentation/liferay-portal/6.1/development/-/ai/lp-6-1-dgen05-available-variables-0>
  + <https://www.liferay.com/es/community/wiki/-/wiki/Main/Access+Objects+from+Velocity>
  + <https://www.liferay.com/es/community/wiki/-/wiki/Main/Custom+Velocity+Variables>

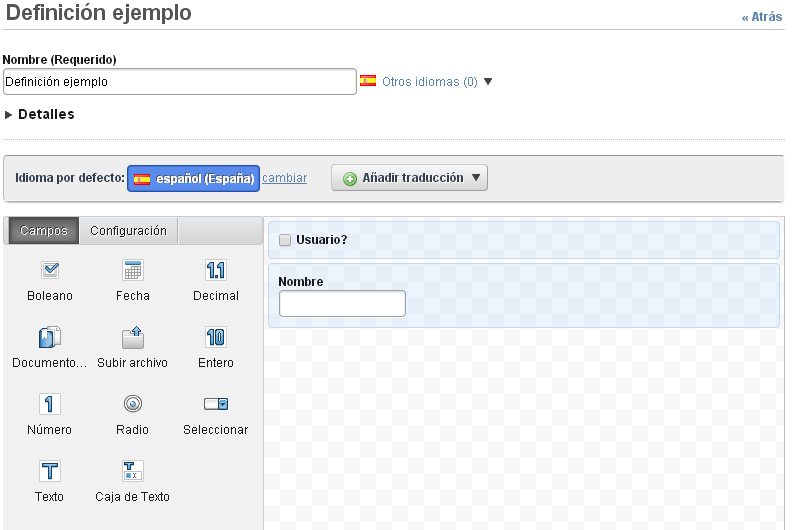
### Listas Dinámicas de Datos

Otra potente herramienta son las DLL o Listas Dinámicas de Datos. A veces queremos tener cierta información no sólo con cierta estructura, sino que queremos tener un listado de ella y mostrarla cuando el administrador de la lista lo requiere. Imaginar por ejemplo el caso de tener que realizar la inscripción de usuarios de éste curso. Para ello tenemos las DLL. Para generarlas, en el panel de control => vista de nuestro sitio => Listas Dinámicas de Datos => Añadir

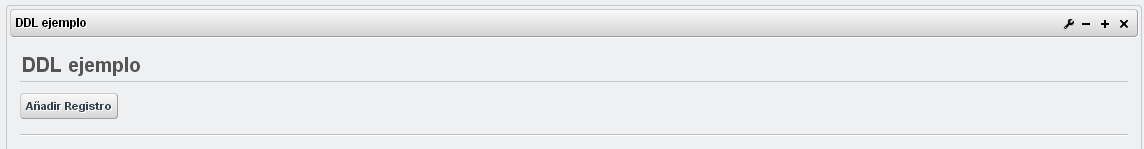


Al crear una nueva tendremos que añadir lo que se llama definición de datos que no son más que ciertos campos que de nuestro listado que como bloque formen un tipo de datos (por ejemplo imaginar uno llamado “Datos Personales” que incluye nombre, apellidos, dirección …).

Creamos una definición de ejemplo con un checkbox y un nombre. Se lo añadimos a la lista dinámica de datos … y listos!



Una vez lo tenemos, para visualizarlo nos vamos a la página en la que queramos incorporarlo y lo incluimos. El componente se llama “Visor de Listas Dinámicas de Datos”. Seleccionamos nuestra lista recién creada.



Si le damos a “Añadir Registro” podemos introducir tantas veces como queramos la información. Ésta información se nos mostrará en forma de lista (incluso podemos seleccionar en la configuración una vista en forma de Excel). Lo normal será querer cambiar ésta vista. Para ello pulsamos sobre “Añadir Plantilla de Lista” y, al igual que con las estructuras, podemos crear en el lenguaje que queramos la plantilla para el listado.

A continuación se define un ejemplo realizado con velocity.

<h1>Listado inscritos</h1>

A continuación está el listado de registros para nuestra Lista Dinámica de Datos "$reserved\_record\_set\_name.data".

#set ($ddlRecordsUtil = $serviceLocator.findService("com.liferay.portlet.dynamicdatalists.service.DDLRecordLocalService"))

#set ($recordSetId = $getterUtil.getLong($reserved\_record\_set\_id.data))

#set ($records = ${ddlRecordsUtil.getRecords($recordSetId)})

<ul>

#foreach ($record in $records)

#set ($name = $record.getField("texto3770").getValue())

<li><em>${name}</em></li>

#end

</ul>

Que nos mostraría la información de la siguiente forma:



Cómo recomendación de desarrollo, la versión de edición de plantillas de la versión posterior de liferay 6.2 está bastante mejorada y con una edición visual. Además ponen a nuestra disposición bastantes variables del portal para su uso así como los campos de la propia lista dinámica de datos.

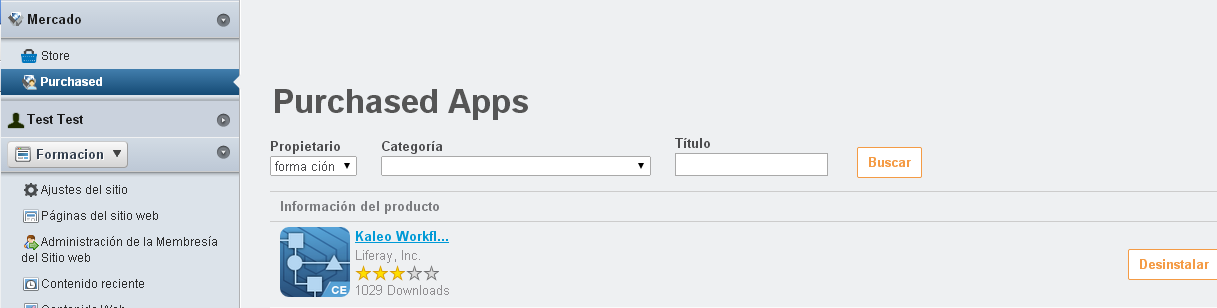
## Workflow

A veces cuando queremos introducir un contenido o queremos que otros lo introduzcan tenemos que tener un control sobre qué y cuándo se publica. Para ello Liferay nos da una integración con Workflow. Con los plugins adecuados del Marketplace podremos utilizar tanto Kaleo, como Activiti como JBPM. Para éste ejemplo utilizaremos Kaleo.

Primero tendremos que descargarnos el Plugin del Marketplace para Kaleo, versión CE para Liferay 6.1.X. En el panel de control existe una opción de Marketplace que nos muestra los diversos plugins disponibles de forma pública. Para conectarnos y descargarnos el que queremos podemos usar los siguientes credenciales del curso:

Usuario : [formacionhcuva@sms.es](mailto:formacionhcuva@sms.es)

Contraseña : formacion

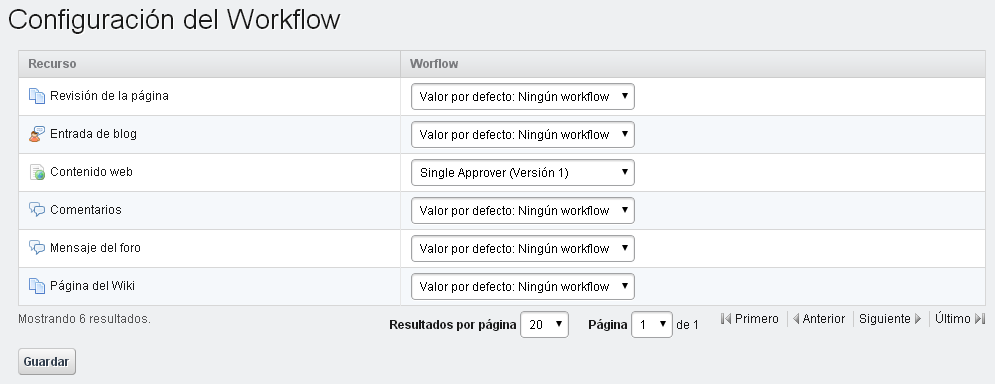


Por defecto ésta acción nos generará lo siguiente:

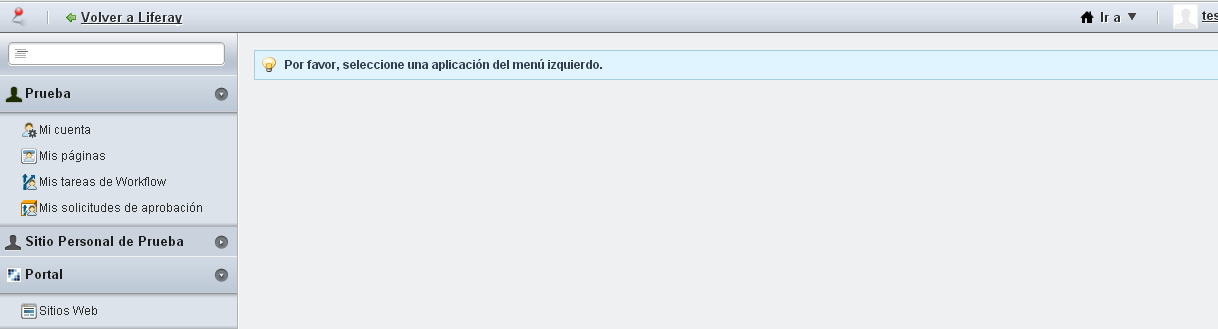
* Un motor de workflow junto a las pestañas de gestión a nivel usuario, sitio, portal …
* Un flujo de validación “single-approver-definition.xml” (adjunto a éste documento) explicado en el curso y que sirve como muestra de lo que puede hacer un flujo de validación.
* Roles content reviewer autogenerados por el flujo anterior. Un rol cuya única función es revisar el contenido/documento/comentario … al que se le asigne y validarlo.

Cómo ejemplo vamos a realizar lo siguiente:

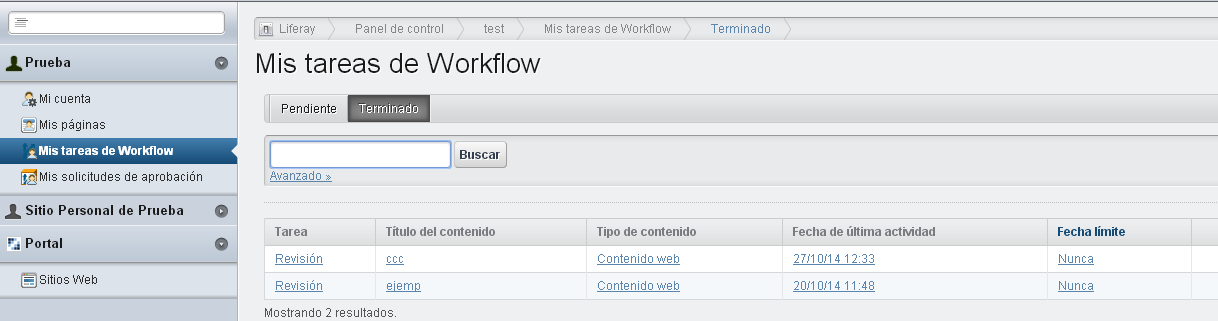
1. Al usuario creado con anterioridad en éste manual como ejemplo le asignamos el rol “Site Content Reviewer”.
2. En el panel de control => vista del sitio “Formación” => “Configuración de Workflow”. Seleccionamos en “Contenido Web” => “Single Approver”



1. Como administrador “test” nos vamos a nuestro sitio e incluimos un contenido web. Ahora no nos saldrá el botón de “publicar”, sino el de “solicitar publicación”. Una vez pulsado el contenido entra en el flujo de validación
2. Ahora accedemos al portal como el usuario creado de ejemplo (en el curso “test2” de tal forma que al acceder al panel de control tendrá dos nuevas opciones.



1. En “Mis tareas de workflow” veremos las tareas que podemos asignarnos o que tenemos asignadas y pendientes de aprobar o rechazar. En “Mis solicitudes de aprobación” tenemos las peticiones que hemos realizado.
2. Si accedemos a “Mis tareas de workflow” podemos ver la petición de revisión del contenido web creado antes por el administrador “test”. Podemos asignárnosla por ejemplo y aprobarla o rechazarla.



1. Finalmente y una vez aprobado ya estará el contenido publicado en la web.

Ésta validación es el caso más simple, uno publica y otro valida. A éste documento se incluyen dos ficheros más de workflow “double-approver-definition.xml” y “paralelo-approver-definition.xml” ambos explicados y usados en el curso.

1. Double-approver-definition.xml: Nos sirve para poner una etapa más en el proceso, necesitando de 2 validaciones para que el contenido sea válido. Además se autogeneran dos roles nuevos como son “Editor” y “Jefe”.
2. Paralelo-approver-definition.xml: Al ver que en la validación anterior, el jefe tiene que esperar hasta que el editor valide para validar él, hemos realizado éste ejemplo para reducir las esperas. Lo que se utiliza son los tags <fork> y <join> para realizar validaciones en paralelo y evitando así esperas. En éste caso un contenido también necesita de la validación tanto del editor como del jefe, pero el segundo no tendrá que esperar al primero para validarlo.

# Entorno Desarrollo, Configuración Eclipse

A continuación vamos a configurar nuestro entorno de desarrollo. Primero nos descargamos la versión de Eclipse que queramos. En el curso se acompaña con la versión Eclipse JEE Luna.

* Descomprimimos Eclipse en la carpeta Formación
* Creamos la carpeta /Workspace
* Al lanzar Eclipse señalamos la carpeta anterior como la carpeta de trabajo.

## Instalación Liferay IDE y Liferay SDK

Liferay nos proporciona un entorno basado en Eclipse para trabajar con las diferentes funcionalidades que nos proporciona. Para ello en Eclipse lo instalamos:

* + Help > Install New Software
  + Añadimos el repositorio de nombre Liferay IDE y url <http://releases.liferay.com/tools/ide/latest/stable/>
  + De las opciones que nos muestran, marcamos Liferay IDE (el resto se verán en un curso avanzado de desarrollo).
  + Aceptar los acuerdos de términos de uso y firma que nos pida.
  + Tras reiniciar Eclipse ya tenemos nuestro entorno con una vista personalizada para Liferay:

Window > Open Perspective > Other > Liferay

* + Por último configuramos el SDK: Window > Preferences > Liferay > Installed Plugin SDKs y añadimos Liferay SDK 6.1.2 descargado. En el curso se acompaña ya con una versión.

## Configuración servidor

Tras configurar nuestro entorno, vamos a ver cómo incluimos nuestro servidor (que viene en el Bundled descargado con anterioridad). Para ello vamos a:

* Pestaña Servers > Define new Server > Liferay 6.1 CE > Next
* Señalamos la carpeta del Tomcat de nuestro Bundled y la versión de Java ya instalada > Next
* Opcionalmente podemos incorporar los zips del código fuente, javadoc y la versión bundled. Todos ellos descargables y también incluidos en la carpeta de formación del curso.

## Primera puesta en marcha

Una vez configurado todo lo anterior ya podemos lanzar nuestro portal en la pestaña Servidores. Podemos observar que se nos muestra lo mismo que cuando lo lanzamos sin Eclipse.

# Debug código fuente

A veces es necesario para el desarrollador saber qué hace Liferay en alguna acción. En nuestros plugins va a ser fácil realizarlo incluyendo breakpoints y lanzando en modo debug el portal, pero para el código fuente de Liferay realizaremos la siguiente configuración:

* New > Other > General > Project. Lo nombramos “Portal”. Quitamos “Use default location” y señalamos el paquete descomprimido liferay-portal-src, descargable desde la web y que se acompaña con el curso.
* En las propiedades del Servidor > Open launch Configuration > Source > Add > Java Project, añadimos el proyecto que acabamos de añadir.

Vamos a ver un ejemplo de qué pasa cuando nos logueamos en el portal :

* Navigate > Go To > Resource > Buscar “UserLocalServiceImpl”. Poner breakpoint en "AuthenticateByEmailAddress”
* Lanzamos en modo debug el portal e intentamos acceder con la cuenta de administrador. Vemos que al acceder, Eclipse nos pregunta si queremos pasar al modo debug. En él podemos ver las variables y el estado de la ejecución del Portal.

# Cambio a base de datos MySql

Cómo sabemos Liferay viene por defecto con una base de datos Hypersonic que es recomendable sólo para desarrollos locales. Para ello vamos a aprender a modificar nuestra instalación para que conecte con una base de datos Mysql. Los pasos son:

* Si no tenemos ya una base de datos instalada y configurada, nos descargamos una. En el curso se acompaña de una lista para descomprimir y un fichero para lanzarla.
* Tras ello tenemos que crear el esquema. Si no tenemos un administrador tipo “Toad for MySQL”, lo realizaremos por consola. Abrimos dos ventanas de comandos “cmd” y realizamos:
  + Lanzar la base de datos: En la carpeta /bin de MySQL escribimos :”mysqld --console”
  + En la otra ventana nos conectaremos con la base de datos y crearemos el esquema básico: “mysql --user=root --password=”. A continuación: “create database lportal;” y salimos: “exit;”
* Ahora en los ficheros de nuestro bundled nos vamos a la ruta donde está nuestro bundled “liferay-portal-6.1.2-ce-ga3” y creamos un fichero “portal-ext.properties”. Éste fichero originalmente estaba en %TOMCAT\_HOME%/webapps/ROOT/WEB-INF/classes pero ya se puede poner en cualquier carpeta o subcarpeta de nuestro Liferay.
* A continuación editamos el fichero creado y ponemos las siguientes características:

jdbc.default.driverClassName=com.mysql.jdbc.Driver

jdbc.default.url=jdbc:mysql://localhost/lportal?useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false

jdbc.default.username=root

jdbc.default.password=

* Al lanzar ahora nuestra instalación de Liferay desde Eclipse, podremos ver como detecta la nueva base de datos. Además autogenera las tablas necesarias que podremos ver con nuestro administrador de base de datos.

# Tipos de Plugins

Existen diferentes tipos de plugins que podemos realizar con Liferay SDK, ya sea por consola o por el IDE de Eclipse. Veremos ejemplos de cada uno más adelante en la formación, pero tener en cuenta que existen actualmente:

* Portlets, Desarrollos con funcionalidades propias
* Themes, Apariencia
* Layouts, Distribución
* Webs, no se verá en el curso, pero sirve para integrar con ciertas aplicaciones web
* Hooks, Extiende funcionalidad
* Ext-Plugin, Caso de última instancia de desarrollo ya que no es reversible al contrario que los Hooks. Llegan donde ningún otro tipo de Plugins pueden llegar.

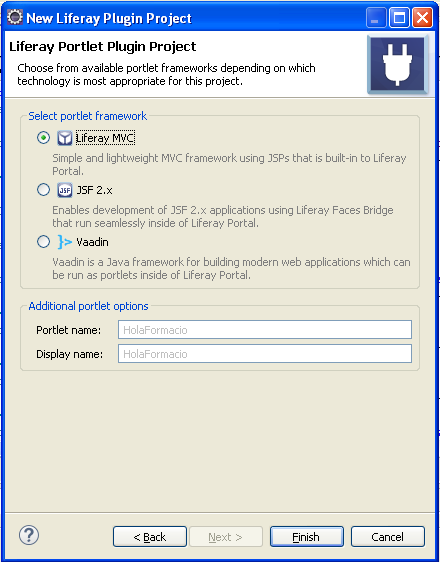
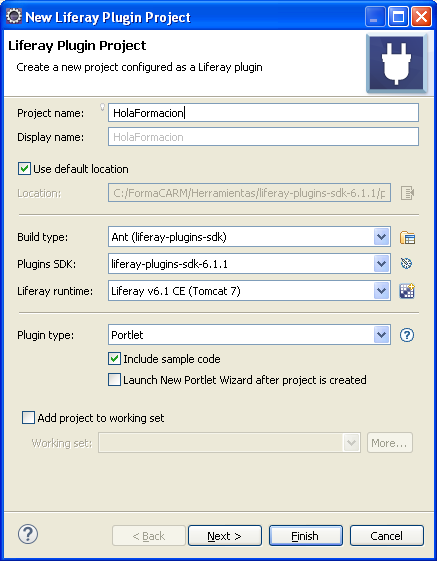
# Desarrollo de Plugins con Liferay IDE

A continuación vamos a ir realizando múltiples ejemplos con la mayoría de los plugins anteriores.

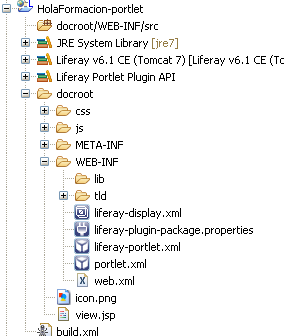
## Primer ejemplo: Hola Formación

Un ejemplo muy básico de cómo desarrollar un portlet mediante el nuevo entorno que tenemos fácilmente sería:

* “New Liferay Plugin Project”
* Nombre : HolaFormacion
* PluginType : Portlet > Next
* Liferay MVC



Observar bien la estructura de portlet que nos ha generado así como los ficheros de configuración. Añadir el portlet arrastrándolo a la pestaña Server y ver cómo se despliega.



En el frontal del portal, nos logueamos y añadimos dicho portlet a la página que queramos. Estará en la categoría “Sample”. Veremos que ya tenemos fácilmente en el Portal lo que hemos indicado en “view.jsp” y en general en nuestro plugin.

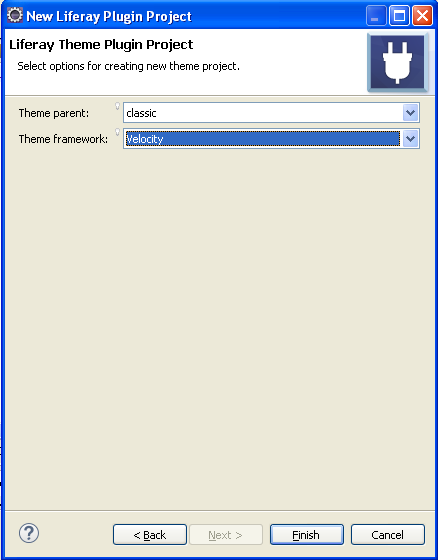
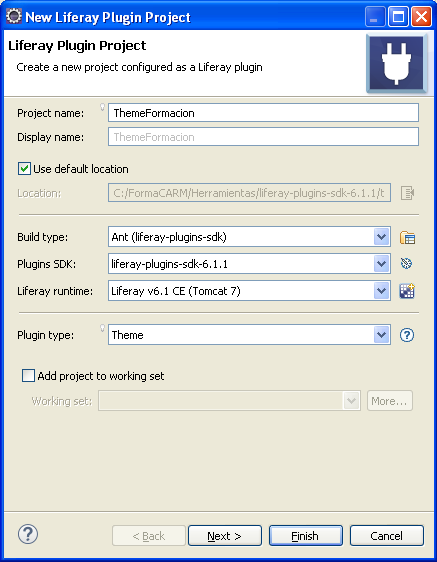
Cómo prueba se puede observar que sin parar el servidor, podemos realizar modificaciones en el desarrollo y actualizando la página ya los tendremos autodesplegados.

Por último comentar que el código se añade al /webapps de nuestro Tomcat para su uso, pero también existe el código del proyecto en la carpeta liferay-sdk/portlets de nuestro SDK.

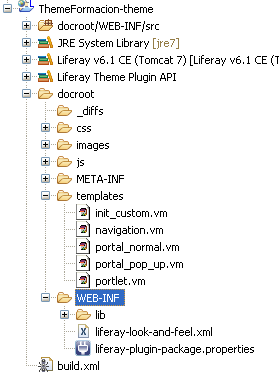
## Creación de un Theme

Un Theme es un diseño preparado y adaptado para Liferay. A través del IDE de Liferay podemos generar uno muy rápido con los siguientes pasos:

* Creamos un nuevo plugin “New Liferay Plugin Project”
* Nombre del proyecto: “FormacionTheme”
* Plugin Type: “Theme” > Next
* Theme parent: classic
* Theme framework: velocity



Mirar la estructura que se nos ha creado de ficheros y prestar especial atención a la carpeta \_diffs que es donde trabajaremos principalmente. Es aquí donde copiaremos los ficheros que queremos modificar del theme parent indicado, siempre respetando la misma estructura.



Primero para que veamos por defecto lo creado, desplegamos nuestro diseño y logeados como administrador, en editar página lo seleccionamos para visualizarlo.

Vemos que el diseño es bastante sencillo e igual al clásico que ya conocíamos. Vamos a ver un par de ejemplos de cómo personalizarlo:

* Para quitar el dockbar, copiamos en la carpeta \_diffs/templates el fichero portal\_normal.vm
* Quitamos la línea siguiente

#if ($is\_signed\_in)

#dockbar()

#end

,lo único que indica ese código es que si está logeado le muestre el dockbar.

* Actualizamos la página y vemos cómo nos ha desaparecido el dockbar cuando estamos logeados en el portal.
* Parecido podemos hacer con los colores. Copiamos el custom.css a \_diffs/css y modificamos el color de background:

*.portlet* {

background-color: *green*;

border: *1px* *solid* *#C8C9CA*;

margin: *0* *0* *10px*;

padding: *3px*;

text-align: *left*;

}

* Podemos observar el cambio de color de fondo de todos los portlets que tengamos en la página donde hemos aplciado el estilo nuevo.

Muchas veces es necesario pasar por el Theme que hemos creado, parámetros según dónde se aplique para mostrar dinámicamente cambios en él. Es un ejemplo claro si quisiéramos mostrar diversas cabeceras para diferentes sitios que tengan el mismo tema de apariencia.

Vamos a aprender cómo pasar variables y utilizarlas en nuestro diseño. Para ello editamos el fichero que hay en /WEB-INF/liferay-look-and-feel.xml e introducimos un parámetro por settings de tal forma que quede algo así esa fracción de código :

<?xml version=*"1.0"*?>

<!DOCTYPE look-and-feel PUBLIC "-//Liferay//DTD Look and Feel 6.1.0//EN" "http://www.liferay.com/dtd/liferay-look-and-feel\_6\_1\_0.dtd">

<look-and-feel>

<compatibility>

<version>6.1.2+</version>

</compatibility>

<theme id=*"FormacionThemeclassic"* name=*"FormacionThemeclassic"* >

<settings>

<setting key=*"nombre"* value=*"formacion"* configurable=*"true"*>

</setting>

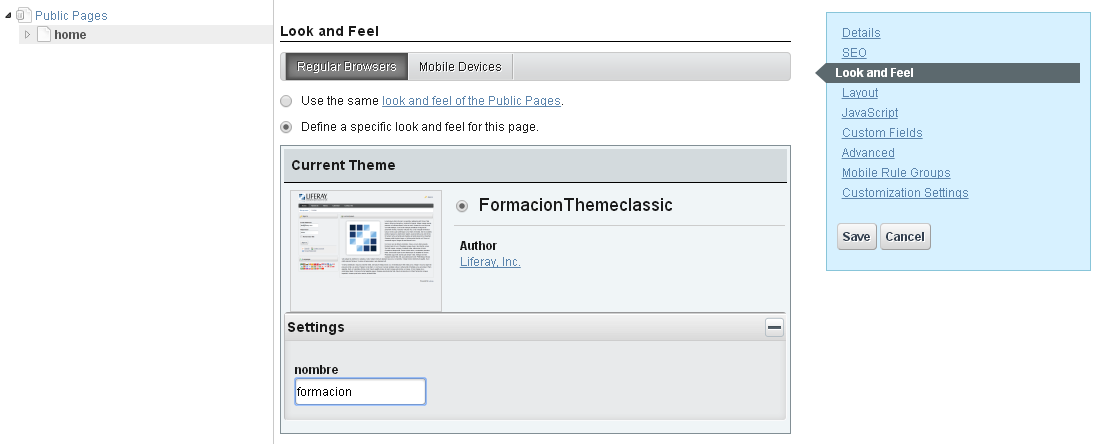
</settings>

</theme>

</look-and-feel>

Lo que indicamos en el código es que va a existir una variable “nombre” de valor inicial “formacion” y configurable desde el panel de control de nuestra página => Apariencia.

Si desplegamos y nos vamos de nuevo a la configuración de la página donde teníamos asignado nuestro nuevo tema de apariencia, veremos que podemos definir un nuevo valor para la variable “nombre”.



Una vez definida vamos a ver cómo podemos referenciarla en nuestro tema. Siguiendo la dinámica anterior, vamos a añadir en nuestro portal\_normal.vml que ya tenemos en la carpeta \_diffs/templates y en la línea en la que se añade el dockbar la cambiamos a lo siguiente:

#if ($is\_signed\_in && $theme.getSetting("nombre") == "formacion")

#dockbar()

#end

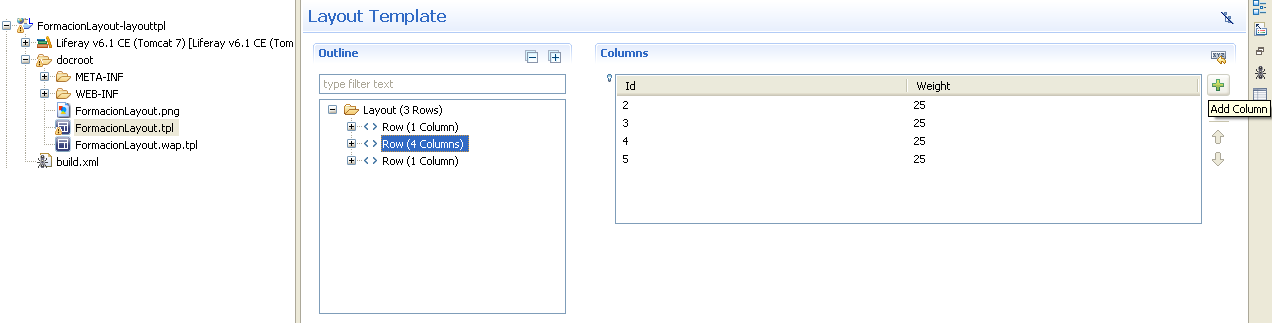
Que le indicamos que además de estar logeado, para que se muestre el dockbar, la variable “nombre” debe valer para esa página “formación”. Guardamos y cuando se nos autodespliegue comprobamos que si cambiamos esa variable a un texto distinto de “formacion” el dockbar no se muestra.

## Creación de un Layout

Para crear una distribución de página, es tan sencillo e intuitivo como:

* Creamos un nuevo plugin “New Liferay Plugin Project”
* Nombre del proyecto: “FormacionLayout”
* Plugin Type: “Layout Template”

Vamos a generar un ejemplo para un layout 1-4-1. Para ello vamos y editamos el fichero “FormacionLayout.tpl”, vista “Design”. En ella sobre nuestro Layout vamos dándole a botón derecho y añadiendo “rows” y columnas ponemos el porcentaje que va a ocupar cada una. Nos debe quedar algo así:



* Lo desplegamos y lo seleccionamos en nuestra página actual.
* Podemos ver desplazando portlets que tenemos la distribución deseada 1-4-1

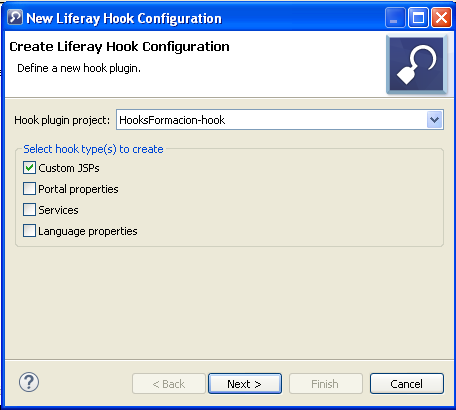
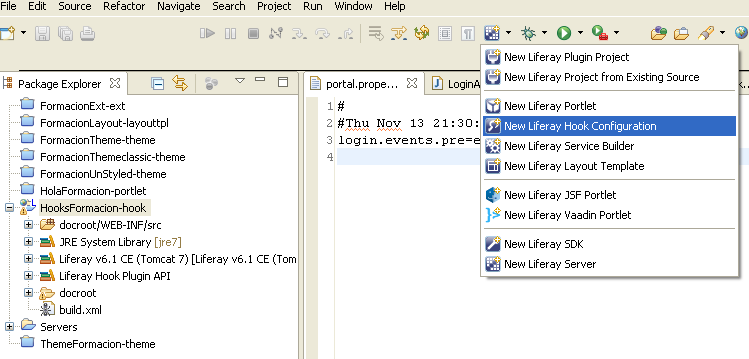
## Creación Hook

Con los hook podemos tocar ficheros, funcionalidades, propiedades … de Liferay sin tener que tocar código original de forma totalmente reversible. Haremos múltiples ejemplos rápidos en éste curso para que podamos observar la potencia de éste tipo de Plugins.

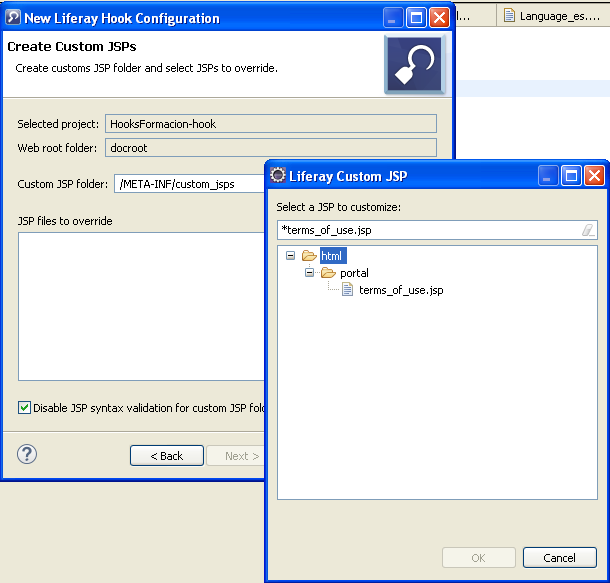
### Modificar JSP Original

Empezaremos por tocar un fichero JSP original del código fuente de Liferay sin modificarlo directamente. Para ello:

* Creamos un nuevo plugin “New Liferay Plugin Project”
* Nombre del proyecto: “FormacionHook”
* Plugin Type: “Hook”
* Creamos un hook “New Liferay Hook Configuration”
* Señalamos nuestro “FormacionHook-hook” y marcamos “Custom JSPs”



* Add From Liferay > /portal/terms\_of\_use.jsp



* + Se nos ha creado el fichero para modificarlo dentro de “/META-INF/custom\_jsps/html/portal/terms\_of\_use.jsp”
* Cambiar “Welcome to our site” por “Bienvenido al curso de Formación”
* Desplegar y crear una nueva cuenta de usuario. Veremos que al mostrarnos los términos de uso, tenemos cambiado ya el texto que se muestra.

Observar a nivel de ficheros, cómo afecta un hook. Si nos vamos a la url %TOMCAT\_HOME%/webapps/ROOT/html/portal veremos que tendremos un terms\_of\_use.jsp, que es nuestro jsp modificado y otro terms\_of\_use.portal.jsp que es el jsp original de Liferay.

Si hacemos un undeploy del portlet veremos que el fichero original vuelve a su nombre inicial.

### Añadir texto a un jsp

Quizás, dado el caso anterior, queramos incluir el jsp y realizarle una moficación general o simplemente tenerlo en un String para añadirle información. Para ello cambiamos todo el texto de nuestro “terms\_of\_use.jsp” por lo siguiente.

<%@ taglib uri=*"http://liferay.com/tld/util"* prefix=*"liferay-util"* %>

<%@ page import=*"com.liferay.portal.kernel.util.StringUtil"* %>

<liferay-util:buffer var=*"html"*>

<liferay-util:include page=*"/html/portal/terms\_of\_use.portal.jsp"* />

</liferay-util:buffer>

<%

html = StringUtil.add(

"Texto Antes",

html,

"\n");

html = StringUtil.add(

html,

"Texto Después",

"\n");

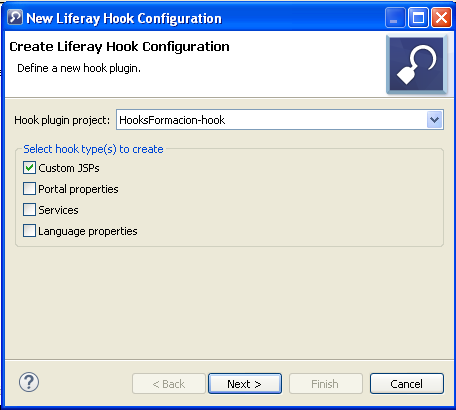
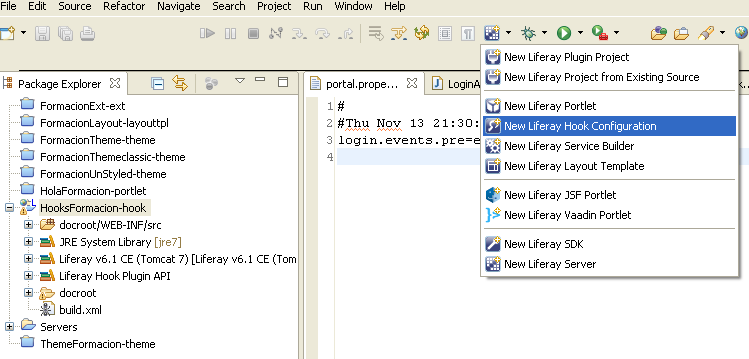
%>

Y vemos que si desplegamos, creamos un usuario y vamos a los términos de uso tenemos el texto original más el añadido.

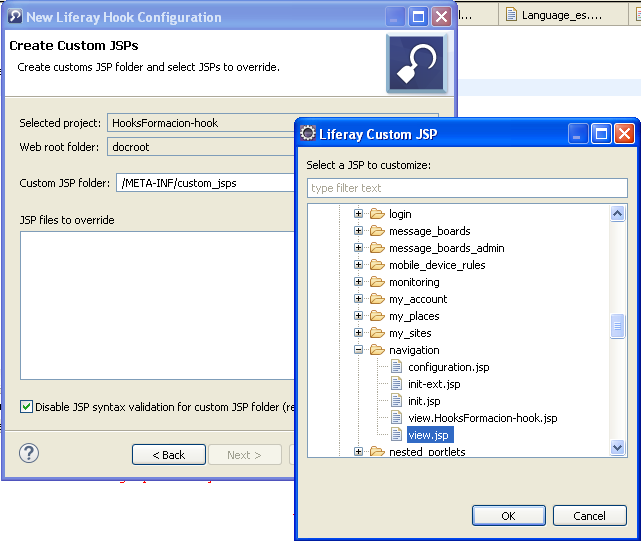
### Modificar JSPs para diferentes sitios

Es posible que muchas veces queramos modificar algún fichero JSP de algún portlet que viene con Liferay por defecto. Además queremos que esos cambios podamos elegir sobre qué sitios se tienen que aplicar. Para ello haremos uso de “Application Adapter”. Haremos

* Plugin Type: “Hook”
* Creamos un hook “New Liferay Hook Configuration”
* Señalamos nuestro “FormacionHook-hook” y marcamos “Custom JSPs”



* Add From Liferay > /portlet/navigation/view.jsp



* + Se nos ha creado el fichero para modificarlo dentro de “/META-INF/custom\_jsps/html/portlet/navigation/view.jsp”

Lo borramos y añadimos lo siguiente:

<%@ taglib uri=*"http://liferay.com/tld/util"*

prefix=*"liferay-util"* %>

<liferay-util:include

page=*"/html/portlet/navigation/view.jsp"*

useCustomPage=*"false"*

/>

<p>

Modificado por Hook y Application Adapter.

</p>

Y en el fichero “liferay-hook.xml” añadimos lo siguiente:

<custom-jsp-dir>**/META-INF/custom\_jsps**</custom-jsp-dir>

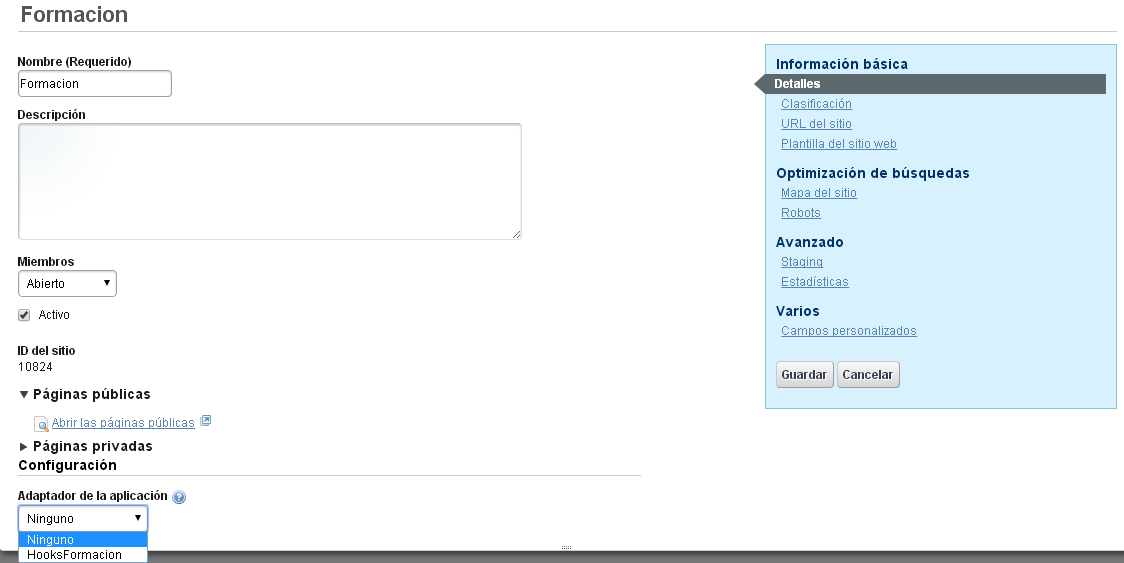
<custom-jsp-global>false</custom-jsp-global>

,de tal forma que a partir de ahora todo jsp modificado en éste hook no se considerará global, sino que estará ligado al sitio en el que lo indiquemos. Si lo desplegamos podemos ver que ahora los ficheros desplegados no tienen el comentado antes \*.portal.jsp, sino que lo particulariza con nuestro código fuente:

\*view.jsp => para el original

\* view.HooksFormacion-hook.jsp => para nuestro jsp modificado.

Ahora nos vamos a nuestro sitio y en la configuración de él tendremos la nueva opción:



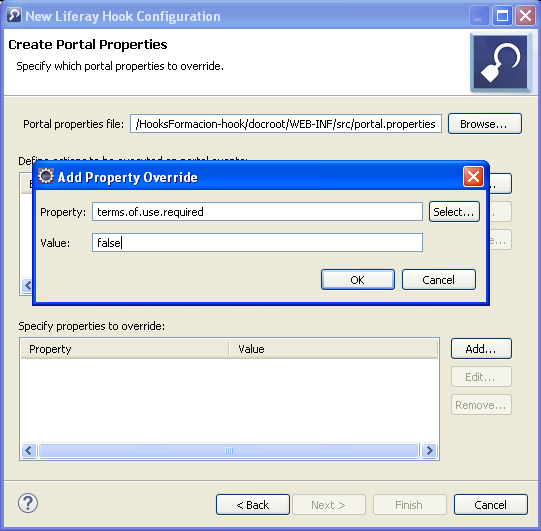
,y así en nuestro sitio veremos el navegador modificado, pero sólo para éste, no para el resto de sitios del portal en el que tengamos añadido dicho portlet.



### Modificar Propiedades del Portal

Ahora vamos a ver cómo podemos modificar propiedades de Liferay. Para ello:

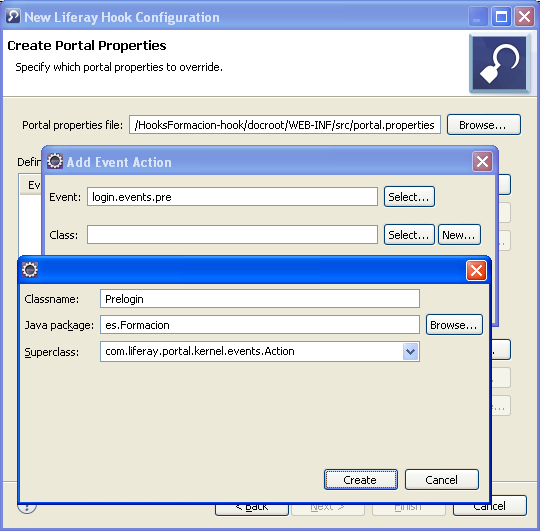
* Creamos un hook “New Liferay Hook Configuration”
* Señalamos nuestro “FormacionHook-hook” y marcamos “Portal Porperties”
* En propiedades: Add > Property > terms.of.use.required . Value: false
* Creamos una nueva cuenta y comprobamos que ahora no nos pide los términos de uso.



### Incluir Acciones a Eventos del Portal

También podemos realizar acciones antes o después de algunos eventos determinados. Para ello hacemos:

* Creamos un hook “New Liferay Hook Configuration”
* Señalamos nuestro “FormacionHook-hook” y marcamos “Portal Porperties”
* En eventos: Add > Event Select > login.events.pre.
* Class: new. Classname: Prelogin; Java Package: es.formacion; Superclass: com.liferay.portal.kernel.events.Action



* En la clase creada, el método “run” añadimos:

System.out.println(“Formación: Pre login action”);

package es.formacion;

import com.liferay.portal.kernel.events.Action;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.liferay.portal.kernel.events.ActionException;

public class PreLogin extends Action {

/\* (non-Java-doc)

\* @see com.liferay.portal.kernel.events.Action#Action()

\*/

public LoginAction() {

super();

}

/\* (non-Java-doc)

\* @see com.liferay.portal.kernel.events.Action#run(HttpServletRequest request, HttpServletResponse response)

\*/

public void run(HttpServletRequest request, HttpServletResponse response) throws ActionException {

System.out.println("Me he logeado! " + request.getLocalName());

}}

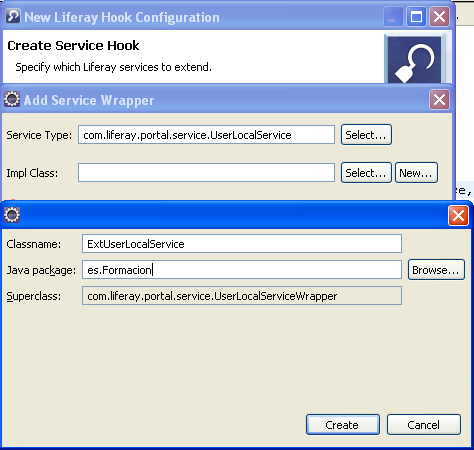
* Tras autodesplegarse, hacemos login con un usuario y podremos ver la marca que deja en la consola. También podríamos hacer un debug fácilmente poniendo un breakpoint.

### Extender los Servicios de Liferay

Cómo se explica durante el curso, todas la acciones principales de Liferay están tratadas como Servicios, ya sean tanto locales como remotos. Por ejemplo para las acciones que tienen que ver con los usuarios, tenemos “UserLocalService” para servicios remotos o “UserService” para invocación remota. Nunca extenderemos o atacaremos éstas interfaces directamente, siempre que usaremos la clase que las enmascara, del mismo nombre con el sufijo “Wrapper”, por ejemplo “UserLocalServiceWrapper”.

Para crear un servicio:

* Creamos un hook “New Liferay Hook Configuration”
* Señalamos nuestro “FormacionHook-hook” y marcamos “Services”
* En servicios para extender: Add
* Service Type : UserLocalService
* Classname: ExtUserLocalService; Java Package: es.formacion; Superclass: com.liferay.portal.service.UserLocalServiceWrapper



Sobre la clase generada hacemos una sobreescritura del método “GetUserById” y ponemos una marca para ver que cada vez que ese servicio es invocado, podemos añadir nuestro código.

package es.formacion;

import com.liferay.portal.kernel.exception.PortalException;

import com.liferay.portal.kernel.exception.SystemException;

import com.liferay.portal.model.User;

import com.liferay.portal.service.UserLocalServiceWrapper;

import com.liferay.portal.service.UserLocalService;

public class ExtUserLocalService extends UserLocalServiceWrapper {

/\* (non-Java-doc)

\* @see com.liferay.portal.service.UserLocalServiceWrapper#UserLocalServiceWrapper(UserLocalService userLocalService)

\*/

public ExtUserLocalService(UserLocalService userLocalService) {

super(userLocalService);

}

@Override

public User getUserById(long userId) throws PortalException,

SystemException {

// TODO Auto-generated method stub

System.out.println("Sobreescribiendo servicios " + userId);

return super.getUserById(userId);

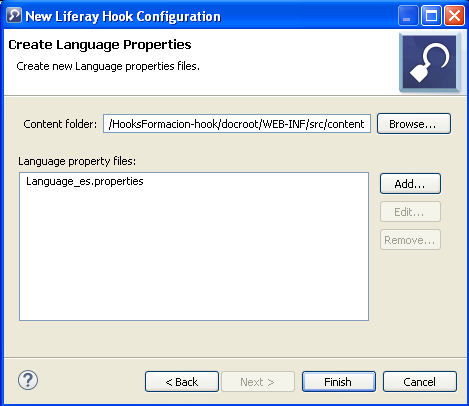
}

}

### Modificación de ficheros de traducción

Por último en los hooks veremos como tocar la traducción de los ficheros de idiomas que Liferay nos proporciona.

* Creamos un hook “New Liferay Hook Configuration”
* Señalamos nuestro “FormacionHook-hook” y marcamos “Language Properties” > Add > “Language\_es.properties”



* En el fichero que nos genera añadimos:

javax.portlet.title.58=Acceso Usuarios Formacion

control-panel=Configuración Portal Formacion

* Al autodesplegarse y actualizar ( si tenemos el portal en español ) veremos que estos textos han cambiado.



## Creación de Portlets

### Portlet con dos modos

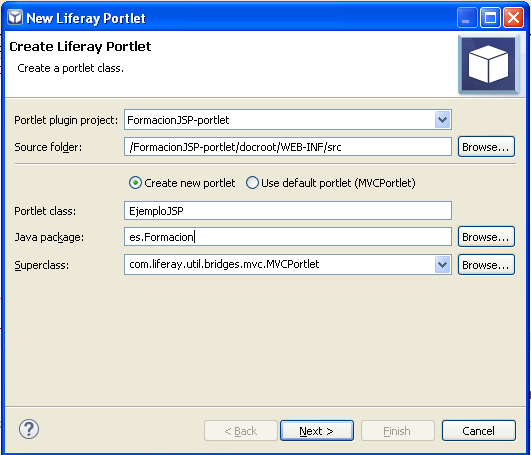
En éste ejemplo básico, tras el hola mundo, vamos a ver un caso de un portlet que contiene dos modos de pantalla: View y Edit. De tal forma que vamos a aprender cómo podemos definir variables en la ventana de edición que afecten a la vista del portlet.

Para ello creamos lo siguiente:

* Creamos un Portlet “New Liferay Plugin Project”
* Lo llamamos “FormacionJSP”
* Plugin Type : Portlet.
* Deseleccionar “Include Sample Code” y seleccionar “Launch New Portlet Wizard …” > Next
* Marcamos Liferay MVC > Finish

En la ventana que se abre indicamos

* Create New Portlet
* Portlet Class : EjemploJSP
* Java Package : es.formacion
* Superclass : com.liferay.util.bridges.mvc.MVCPortlet > Next
* Marcamos en Portlets Modes : Edit también



Entre la estructura generada vamos a añadir lo siguiente:

* En la clase EjemploJSP sobreescribimos el método ProcessAction original poniendo:

@Override

**public** **void** processAction(

ActionRequest actionRequest, ActionResponse actionResponse)

**throws** IOException, PortletException {

PortletPreferences prefs = actionRequest.getPreferences();

String name = actionRequest.getParameter("nombre");

**if** (name != **null**) {

prefs.setValue("nombre", name);

prefs.store();

}

actionResponse.setPortletMode(PortletMode.*VIEW*);

**super**.processAction(actionRequest, actionResponse);

}

* En los JSP autogenerados, añadimos en View.jsp:

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ page import=*"javax.portlet.PortletPreferences"* %>

<portlet:defineObjects />

<% PortletPreferences prefs = renderRequest.getPreferences();

String nombre = (String)prefs.getValue(

"nombre", ""); %>

<p>Bienvenido</p>

<p><%= nombre %>!</p>

* En Edit.jsp:

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/aui"* prefix=*"aui"* %>

<%@ page import=*"javax.portlet.PortletPreferences"* %>

<portlet:defineObjects />

<%

PortletPreferences prefs = renderRequest.getPreferences();

String nombre = (String)prefs.getValue(

"nombre", "");

%>

<form action=*"*<portlet:actionURL />*"* method=*"post"*>

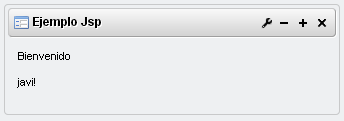
<aui:input label=*"nombre"* name=*"nombre"* type=*"text"* value=*"*<%=

nombre %>*"* />

<aui:button type=*"submit"* />

</form>

Desplegamos y vemos la funcionalidad conseguida. En la opción “Preferencias” indicamos un nombre. Automáticamente en la parte visual podemos ver cómo se muestra el nombre indicado.



Tras ver las fases que tiene un portlet ( Action y Render ) vamos a verlas mediante código, para ello:

* + Comentamos de EjemploJSP.java la línea:

actionResponse.setPortletMode(PortletMode.VIEW);

* + Cambiamos edit.jsp por:

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/aui"* prefix=*"aui"* %>

<%@ page import=*"javax.portlet.PortletPreferences"* %>

<portlet:defineObjects />

<%

PortletPreferences prefs = renderRequest.getPreferences();

String nombre = (String)prefs.getValue(

"nombre", "");

%>

<portlet:actionURL var=*"editURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/ejemplojsp/edit.jsp"* />

</portlet:actionURL>

<!-- <form action="<portlet:actionURL />" method="post"> -->

<form action=*"*<%=editURL %> *%>"* method=*"post"*>

<aui:input label=*"nombre"* name=*"nombre"* type=*"text"* value=*"*<%=

nombre %>*"* />

<aui:button type=*"submit"* />

</form>

<portlet:renderURL var=*"viewURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/ejemplojsp/view.jsp"* />

</portlet:renderURL>

<br/>

<p><a href=*"*<%= viewURL %>*"*>Volver</a></p>

### IPC Portlet

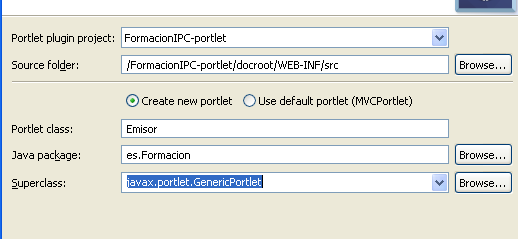
A veces es necesario que dos portlets se comuniquen entre sí. Para ello vamos a realizar un ejemplo que nos servirá para entender mejor cómo realizarlo.

Para ello creamos lo siguiente:

* Creamos un Portlet “New Liferay Plugin Project”
* Lo llamamos “FormacionIPC”
* Plugin Type : Portlet.
* Deseleccionar “Include Sample Code” y seleccionar “Launch New Portlet Wizard …” > Next
* Marcamos Liferay MVC > Finish

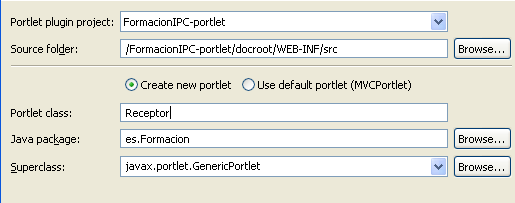
En la ventana que se abre indicamos

* Create New Portlet
* Portlet Class : Emisor
* Java Package : es.formacion
* Superclass : javax.portlet.GenericPortlet > Next



Creamos un segundo portlet

* Create Liferay Portlet
* Portlet Class : Receptor
* Java Package : es.formacion
* Superclass : javax.portlet.GenericPortlet > Next



En Emisor.java añadimos el siguiente método que realiza la acción de enviar al receptor una señal de sorteo:

@ProcessAction(name="lanzarSorteo")

public void lanzarSorteo(ActionRequest request, ActionResponse response) {

String suerte = null;

// Send an Event that the ball has been pitched.

Random random = new Random(System.currentTimeMillis());

int numero = random.nextInt(4) + 1;

switch (numero) {

case 1:

suerte = "Hospital General ";

break;

case 2:

suerte = "Policlínico";

break;

case 3:

suerte = "Urgencias";

break;

case 4:

suerte = "Maternidad";

break;

default:

suerte = "Hospital General";

}

QName qName = new QName("http://sms.es/events", "ipc.sorteo");

response.setEvent(qName, suerte);

}

En Receptor.java recogemos la acción enviada anteriormente:

@ProcessEvent(qname="{http://sms.es/events}ipc.sorteo")

public void RecibirSorteo(EventRequest request, EventResponse response) {

Event event = request.getEvent();

String suerte = (String)event.getValue();

response.setRenderParameter("suerte", suerte);

}

Ahora generamos la parte visual, modificando los JSPs. View.jsp del emisor:

<%@ taglib uri="http://java.sun.com/portlet\_2\_0" prefix="portlet" %>

<portlet:defineObjects />

<p>Pulse el botón para sortear un edificio del SMS</p>

<a href="<portlet:actionURL name="lanzarSorteo"></portlet:actionURL>">Sortear!</a>

Y ahora el View.jsp del receptor:

<%@ taglib uri="http://java.sun.com/portlet\_2\_0" prefix="portlet" %>

<portlet:defineObjects />

<%

String suerte = (String)renderRequest.getParameter("suerte");

%>

<p>Y el edificio es ...</p>

<p>

<% if (suerte!=null) { %>

<%=suerte %>!

<% } else { %>

... Esperando sorteo.

<% } %>

También tendremos que modificar dos ficheros de configuración. En liferay-plugin-package.properties ponemos al final las librerías que son necesarias para que funcione correctamente:

portal.dependency.jars=\

jstl.jar,\

jstl-impl.jar

En el fichero portlet.xml añadimos las sentencias para permitir que ambos portlets soporten los eventos. Antes del fin de </portlet> del emisor añadimos:

<supported-publishing-event>

<qname xmlns:x=*"http://sms.es/events"*>x:ipc.sorteo</qname>

</supported-publishing-event>

,antes del fin de </portlet> del receptor hacemos algo parecido:

<supported-processing-event>

<qname xmlns:x=*"http://sms.es/events"*>x:ipc.sorteo</qname>

</supported-processing-event>

Y por último antes de terminar el tag </portlet-app> añadimos la definición del evento

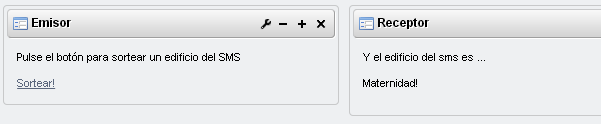
<event-definition>

<qname xmlns:x=*"http://sms.es/events"*>x:ipc.sorteo</qname>

<value-type>java.lang.String</value-type>

</event-definition>

Una vez hemos comprobado que todo está correcto, añadimos ambos portlets a una página y vemos como se comunican entre sí al pulsar el botón de sortear en el portlet Emisor.



*Ejercicio propuesto: que el receptor también emita y emisor reciba.*

*Solución:*

En Emisor.java añadimos el siguiente método para recibir ahora:

@ProcessEvent(qname="{http://sms.es/events2}ipc.sorteo2")

**public** **void** RecibirSorteo2(EventRequest request, EventResponse response) {

Event event = request.getEvent();

String suerte = (String)event.getValue();

response.setRenderParameter("suerte2", suerte);

}

En Receptor.java añadimos el método para enviar:

@ProcessAction(name="lanzarSorteo2")

**public** **void** lanzarSorteo2(ActionRequest request, ActionResponse response) {

String suerte = **null**;

// Send an Event that the ball has been pitched.

Random random = **new** Random(System.*currentTimeMillis*());

**int** numero = random.nextInt(4) + 1;

**switch** (numero) {

**case** 1:

suerte = "Hospital General";

**break**;

**case** 2:

suerte = "Policlínico";

**break**;

**case** 3:

suerte = "Urgencias";

**break**;

**case** 4:

suerte = "Maternidad";

**break**;

// should never print

**default**:

suerte = "Hospital General";

}

QName qName = **new** QName("http://sms.es/events2", "ipc.sorteo2");

response.setEvent(qName, suerte);

}

Ahora generamos la parte visual, modificando los JSPs. View.jsp del emisor:

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<portlet:defineObjects />

<p>Pulse el botón para sortear un edificio del SMS</p>

<%

String suerte = (String)renderRequest.getParameter("suerte2");

%>

<p>Y el edificio del sms de vuelta es ...</p>

<p>

<% **if** (suerte!=**null**) { %>

<%=suerte %>!

<% } **else** { %>

... Esperando sorteo.

<% } %>

<a href=*"*<portlet:actionURL name=*"lanzarSorteo"*></portlet:actionURL>*"*>Sortear!</a>

Y ahora el View.jsp del receptor:

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<portlet:defineObjects />

<%

String suerte = (String)renderRequest.getParameter("suerte");

%>

<p>Y el edificio del sms es ...</p>

<p>

<% **if** (suerte!=**null**) { %>

<%=suerte %>!

<% } **else** { %>

... Esperando sorteo.

<% } %>

<a href=*"*<portlet:actionURL name=*"lanzarSorteo2"*></portlet:actionURL>*"*>Sortear!</a>

En el fichero portlet.xml añadimos las sentencias para permitir que ambos portlets soporten los eventos. Antes del fin de </portlet> del emisor tendremos de antes y ahora:

<supported-processing-event>

<qname xmlns:x=*"http://sms.es/events2"*>x:ipc.sorteo2</qname>

</supported-processing-event>

<supported-publishing-event>

<qname xmlns:x=*"http://sms.es/events"*>x:ipc.sorteo</qname>

</supported-publishing-event>

, y antes del fin de </portlet> del receptor hacemos algo parecido:

<supported-processing-event>

<qname xmlns:x=*"http://sms.es/events"*>x:ipc.sorteo</qname>

</supported-processing-event>

<supported-publishing-event>

<qname xmlns:x=*"http://sms.es/events2"*>x:ipc.sorteo2</qname>

</supported-publishing-event>

Y por último antes de terminar el tag </portlet-app> añadimos el nuevo evento, quedando ambos como:

<event-definition>

<qname xmlns:x=*"http://sms.es/events"*>x:ipc.sorteo</qname>

<value-type>java.lang.String</value-type>

</event-definition>

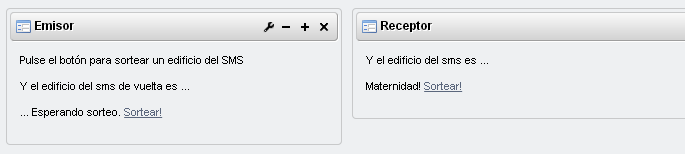
<event-definition>

<qname xmlns:x=*"http://sms.es/events2"*>x:ipc.sorteo2</qname>

<value-type>java.lang.String</value-type>

</event-definition>

Una vez hemos comprobado que todo está correcto, añadimos ambos portlets a una página y vemos como se comunican entre sí en ambos sentidos.



### Ejemplos Portlet PHP

A veces cuando tenemos que migrar un portal existente, nos encontramos con tecnologías que no son la forma habitual de trabajar en Liferay. Un caso común es PHP. Para ello vamos a ver como crear portlets que soportan ésta tecnología para implementar la funcionalidad necesaria.

La forma clásica de desplegar php en Liferay era:

* tener nuestros php con el código
* comprimirlos en un fichero .zip
* instalarlo a través del panel de control mediante el plugin installer => Upload File

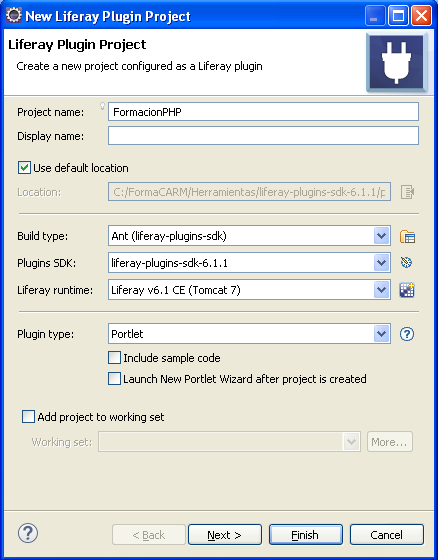
Tras la implementación del Marketplace la mecánica de despliegue de desarrollos cambió por lo que esta opción desapareció para la versión 6.1.X. Aún así existe en el Marketplace el plugin para instalar:

<https://www.liferay.com/es/marketplace/-/mp/application/15474932>

Nosotros vamos a ver una forma de “adaptar” el Liferay IDE existente para que sin necesidad de muchos cambios, podamos tener nuestros desarrollos PHP en nuestro entorno.

Para ello creamos lo siguiente:

* Creamos un Portlet “New Liferay Plugin Project”
* Lo llamamos “FormacionPHP”
* Plugin Type : Portlet.
* Deseleccionar “Include Sample Code” y también deseleccionar “Launch New Portlet Wizard …” > Next
* Marcamos Liferay MVC > Finish



Ahora en nuestro proyecto realizaremos dos funciones:

* Añadir nuestros ficheros PHP necesarios en /docroot de nuestro proyecto.
* Eliminar el fichero “portlet.xml”

Ya podremos añadir nuestro proyecto y se desplegaría. Como ejemplo vamos a añadir el fichero index.php siguiente:

<div>

<b>Portlet PHP</b>

</div>

<br />

Parámetros Post: <?php print\_r($\_POST); ?>

<br /><br />

Parámetros Get: <?php print\_r($\_GET); ?>

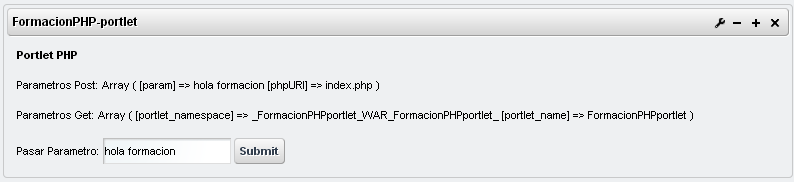
<br /><br />

<form action="index.php" method="post">

Pasar Parámetro: <input name="param" type="text" value="<?php echo $\_GET["param"]?$\_GET["param"]:$\_POST["param"]; ?>">

<input type="submit" value="Submit" />

Desplegamos y en la categoría “Indefinido” tendremos nuestro portlet:



Liferay utiliza para esto una tecnología llamada Quercus, que basada en Java interpreta PHP.

Para ver cómo podemos desde Quercus / Liferay recoger algunas variables del Portal, vamos a realizar el siguiente ejemplo. En nuestro index.php añadimos debajo del todo, antes del cierro del form </form>:

<?php include("navigation.php"); ?>

A continuación creamos los siguientes ficheros al mismo nivel que éste:

\* navigation.php

<div class="separator"></div>

<div>

<a href="index.php?param=formacion">Home</a> -

<a href="portlet\_url.php">Llamada a Portlet</a> -

<a href="test\_db.php">Test conexion a base de datos</a>

</div>

, que nos hará de navegación entre páginas

\* portlet\_url.php

<?php

import com.liferay.portal.kernel.util.WebKeys;

import com.liferay.portal.util.PortletKeys;

import com.liferay.portlet.PortletURLFactoryUtil;

import javax.portlet.PortletMode;

import javax.portlet.PortletRequest;

import javax.portlet.WindowState;

$request = quercus\_servlet\_request();

$themeDisplay = $request->getAttribute(WebKeys::THEME\_DISPLAY);

$myAccountURL = PortletURLFactoryUtil::create(

$request, PortletKeys::MY\_ACCOUNT, $themeDisplay->getPlid(),

PortletRequest::RENDER\_PHASE);

$myAccountURL->setParameter("struts\_action", "/my\_account/edit\_user");

$myAccountURL->setPortletMode(PortletMode::VIEW);

$myAccountURL->setWindowState(WindowState::MAXIMIZED);

?>

<a href="<?php echo $myAccountURL->toString(); ?>">Mi Cuenta</a>

<?php include("navigation.php"); ?>

,que nos indicará como a través de Quercus podemos recoger el request y realizar las acciones normales sobre él como recoger variables tipo Theme\_Display. También vemso como se puede incluir un portlet en la misma ventana que estamos.

\* test\_db.php

<?php

import com.liferay.portal.kernel.dao.db.DB;

import com.liferay.portal.kernel.util.PropsUtil;

import com.liferay.portal.kernel.util.StringUtil;

$url = PropsUtil::get("jdbc.default.url");

$dbType = StringUtil::split($url, ":")[1];

$host = StringUtil::split($url, "/")[2];

$username = PropsUtil::get("jdbc.default.username");

$password = PropsUtil::get("jdbc.default.password");

if ($dbType == DB::TYPE\_MYSQL) {

$link = mysql\_connect($host, $username, $password);

}

if ($link) {

echo "Se ha conectado correctamente a la base de datos tipo $dbType.";

}

else {

echo "No se ha podido conectar con la base de datos tipo $dbType.";

}

?>

<?php include("navigation.php"); ?>

,que nos enseña como podemos recoger variables del fichero de propiedades de liferay a través de las llamadas normales a PropsUtil.

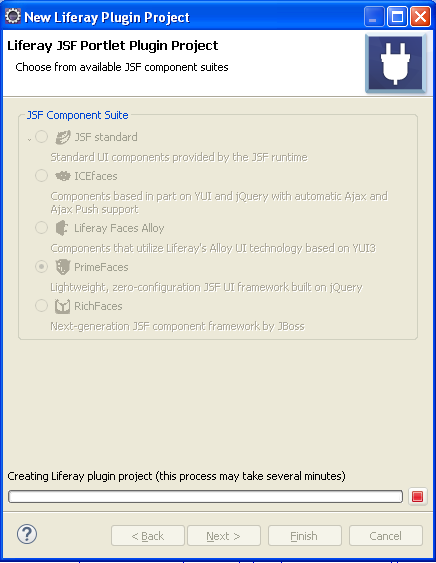
### Ejemplo Portlet JSF Bridge

Como reseña, vamos a ver cómo se crea fácilmente un portlet que soporte JSF gracias al proyecto de Liferay JSF Bridge y que nos genera automáticamente toda la capa de compatibilidad entre un portlet y un desarrollo JSF.

Realizaremos un ejemplo con la librería PrimeFaces, bastante potente en componentes ( [www.primefaces.org](http://www.primefaces.org) )

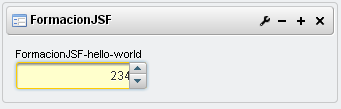
Como siempre seguimos los pasos:

* Creamos un Portlet “New Liferay Plugin Project”
* Lo llamamos “FormacionJSF”
* Plugin Type : Portlet.
* Seleccionar “Include Sample Code” > Next
* Marcamos JSF 2.x > Next
* Entre todas las librerías y frameworks disponibles seleccionamos PrimeFaces



* Para ver un caso práctico en nuestro fichero view.xhtml añadiremos el componente de Primefaces que sirve como un input con botones de incrementar:

<p:spinner />



*Error en eclipse luna:*

[*http://download.eclipse.org/eclipse/updates/4.4/*](http://download.eclipse.org/eclipse/updates/4.4/)

[*https://bugs.eclipse.org/bugs/show\_bug.cgi?id=445122*](https://bugs.eclipse.org/bugs/show_bug.cgi?id=445122)

* *update*
* *all => ant*
* *actulizar el proyecto F5*
* *portlet.xml => duplicado xmlns:schema*

Para un ejemplo más real con JSF, vamos a usar un componente más potente como es el datatable. Para ello añadimos en nuestro view.xhtml:

<p:dataTable var=*"alumno"* value=*"#{TablaBean.alumnos}"*>

<p:column headerText=*"Nombre"*>

<h:outputText value=*"#{alumno.nombre}"* />

</p:column>

<p:column headerText=*"Email"*>

<h:outputText value=*"#{alumno.email}"* />

</p:column>

</p:dataTable>

Y creamos el paquete para nuestras clases “es.formacion”. En él añadimos nuestras clases:

\* Alumno.java

**package** es.formacion;

**public** **class** Alumno {

**private** String nombre;

**private** String email;

**public** String getNombre() {

**return** nombre;

}

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

}

\* TablaBean.java

**package** es.formacion;

**import** java.io.Serializable;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.faces.bean.ManagedBean;

**import** javax.faces.bean.ViewScoped;

@ManagedBean(name="TablaBean")

@ViewScoped

**public** **class** TablaBean **implements** Serializable {

**private** List<Alumno> alumnos;

**public** List<Alumno> getAlumnos() {

alumnos = **new** ArrayList<Alumno>();

Alumno alumno1 = **new** Alumno(); alumno1.setEmail("alumno1@sms.es"); alumno1.setNombre("alumno1");

Alumno alumno2 = **new** Alumno(); alumno2.setEmail("alumno2@sms.es"); alumno2.setNombre("alumno2");

Alumno alumno3 = **new** Alumno(); alumno3.setEmail("alumno3@sms.es"); alumno3.setNombre("alumno3");

Alumno alumno4 = **new** Alumno(); alumno4.setEmail("alumno4@sms.es"); alumno4.setNombre("alumno4");

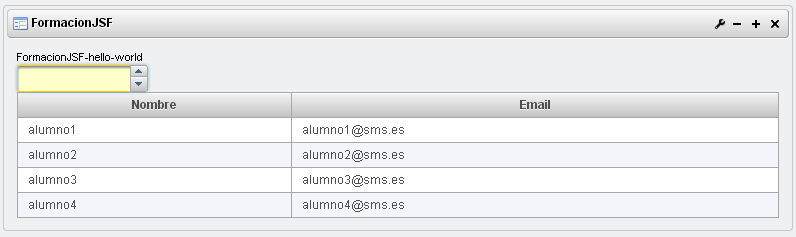
alumnos.add(alumno1); alumnos.add(alumno2); alumnos.add(alumno3); alumnos.add(alumno4);

**return** alumnos;

}

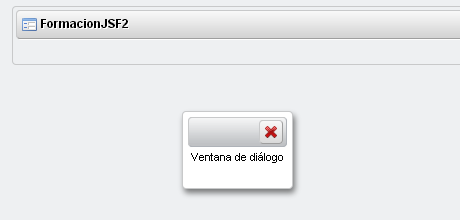
}

,y debemos ver lo siguiente al desplegarlo en nuestro portal:



*Ejercicio propuesto, hacer un portlet JSF con AlloyUI “Formación JSF2”. Añadiendo un cuadro de diálogo con algún texto.*

<aui:dialog><aui:field>Ventana de diálogo</aui:field></aui:dialog>



## Service Builder Portlet

En éste ejemplo vamos a ver cómo a través del Service Builder podemos realizar las capas de modelo, persistencia y servicio de nuestra aplicación.

Imaginemos una aplicación que tengamos un directorio corporativo en el cual tenemos que dar de alta categorías, cargos y servicios y añadir empleados que los contengan. Vamos a ver como automatizar lo máximo posible éste escenario.

### Definir nuestro ORM y configurar Service.xml

Para ello creamos lo siguiente:

* Creamos un Portlet “New Liferay Plugin Project”
* Lo llamamos “FormacionService”
* Plugin Type : Service Builder Portlet.
* Deseleccionar “Include Sample Code” > Finish

Creamos un primero portlet:

* New Liferay Portlet
* Portlet Class : ListadoEmpleados
* Java Package : es.formacion
* Superclass : com.liferay.util.bridges.mvc.MVCPortlet > Finish

Después generamos otros portlets para incluir las Categorías, Cargos y Servicios:

* New Liferay Portlet
* Portlet Class : ListadoCategorias
* Java Package : es.formacion
* Superclass : com.liferay.util.bridges.mvc.MVCPortlet > Finish
* New Liferay Portlet
* Portlet Class : ListadoCargos
* Java Package : es.formacion
* Superclass : com.liferay.util.bridges.mvc.MVCPortlet > Finish
* New Liferay Portlet
* Portlet Class : ListadoServicios
* Java Package : es.formacion
* Superclass : com.liferay.util.bridges.mvc.MVCPortlet > Finish

Si los desplegamos ya en nuestra instalación de Liferay veremos ambos portlets aún sin funcionalidad, salvo la inicial.

Lo primero que haremos es rellenar el modelo de clases y sus atributos. Para ello añadimos en el Service.xml

Parámetros Globales:

* **Package path**: es.formacion
* **Auto namespace tables**: no
* **Author**: HCUVA
* **Namespace**: FormacionService

Es importante el Namespace de aquí tener en cuenta que será el prefijo que le dará a las tablas que autogenerará en la base de datos.

A continuación nuestras entidades, que serán la base de todo:

* **Name:** *Empleado*
* **Local service:** *yes*
* **Remote service:** *yes*
* **Name:** *Cargo*
* **Local service:** *yes*
* **Remote service:** *yes*
* **Name:** *Categoria*
* **Local service:** *yes*
* **Remote service:** *yes*
* **Name:** *Servicio*
* **Local service:** *yes*
* **Remote service:** *yes*

A continuación las columnas que tendrá cada Entidad:

Empleado:

id\_empleado long – clave primaria

nombre – String

apellido1 – String

apellido2 - String

dni – String

Categoria:

id\_categoria – long – clave primaria

desc\_categoria – String

Cargo:

id\_cargo – long – clave primaria

desc\_cargo – String

movil - String

Servicio:

id\_servicio – long – clave primaria

nombre – String

telefono - String

Luego habrá columnas comunes a ambos. Las definimos también ya que son algunas necesarias y otras recomendadas de cara a posibles auditorías.

companyId – long

groupId – long

userId – long

createDate – date

modifiedDate – date

Una vez definidas las entidades, vamos a la pestaña de diagrama y hacemos la relación entre ellas de tal forma que haya una flecha de relación desde Empleado a Cargo, Categorías y Servicios. Esto hará que tengamos nuevas columnas en la entidad Empleado: id\_cargo, id\_categoria, id\_servicio

Una vez obtenidas las entidades, vamos a configurar ya las consultas a base de datos de forma rápida y fácil así como la ordenación de los resultados.

Para ordenar nos vamos al apartado Order de las entidades y definimos por ejemplo:

Empleado : Specify ordering : marcado; By : asc; columna : nombre

Categoria : Specify ordering : marcado; By : asc; columna : desc\_categoria

Cargo : Specify ordering : marcado; By : asc; columna : desc\_cargo

Servicio : Specify ordering : marcado; By : asc; columna : nombre

Y para los “Finders”, especificamos al menos un objeto para cada uno de tal forma que el Servicio nos generará los métodos de fetch, find, remove y count según le indiquemos. Añadimos los siguientes:

Empleado: Nombre “GroupId”, return-type “Collection”. Subapartado columna “groupId”

Categoria: Nombre “GroupId”, return-type “Collection”. Subapartado columna “groupId”

Cargo: Nombre “GroupId”, return-type “Collection”. Subapartado columna “groupId”

Servicio: Nombre “GroupId”, return-type “Collection”. Subapartado columna “groupId”

Una vez ya lo tenemos todo listo, en la pestaña overview, arriba a la derecha de los 3 iconos pulsamos “Build Services”. Nos va a generar una buena cantidad de clases que son las capas de modelo, servicio y persistencia. Realmente de ellas tocaremos muy pocas.

***Documentación recomendada para entender mejor la relación de clases que genera:***

[*https://www.liferay.com/es/documentation/liferay-portal/6.1/development/-/ai/service-builder-generated-code-liferay-portal-6-1-dev-guide-en*](https://www.liferay.com/es/documentation/liferay-portal/6.1/development/-/ai/service-builder-generated-code-liferay-portal-6-1-dev-guide-en)

### Servicios Locales

Para implementar nuestros servicios locales tocaremos las clases \*LocalServiceImpl:

* EmpleadoLocalServiceImpl

**public** Empleado addEmpleado(

**long** userId, **long** groupId, String nombre, String apellido1, String apellido2, String dni,

**long** id\_categoria, **long** id\_cargo, **long** id\_servicio, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

**long** id\_empleado = counterLocalService.increment(Empleado.**class**.getName());

Empleado empleado = empleadoPersistence.create(id\_empleado);

empleado.setNombre(nombre);

empleado.setApellido1(apellido1);

empleado.setApellido2(apellido2);

empleado.setDni(dni);

empleado.setId\_cargo(id\_cargo);

empleado.setId\_categoria(id\_categoria);

empleado.setId\_servicio(id\_servicio);

empleado.setGroupId(groupId);

empleado.setCompanyId(user.getCompanyId());

empleado.setUserId(user.getUserId());

empleado.setCreateDate(serviceContext.getCreateDate(now));

empleado.setModifiedDate(serviceContext.getModifiedDate(now));

**super**.addEmpleado(empleado);

**return** empleado;

}

**public** Empleado deleteEmpleado(Empleado empleado) **throws** SystemException {

**return** empleadoPersistence.remove(empleado);

}

**public** Empleado deleteempleado(**long** empleadoId)

**throws** PortalException, SystemException {

Empleado empleado = empleadoPersistence.findByPrimaryKey(empleadoId);

**return** deleteEmpleado(empleado);

}

**public** Empleado getempleado(**long** empleadoId)

**throws** SystemException, PortalException {

**return** empleadoPersistence.findByPrimaryKey(empleadoId);

}

**public** List<Empleado> getEmpleadosByGroupId(**long** groupId)

**throws** SystemException {

**return** empleadoPersistence.findByGroupId(groupId);

}

**public** List<Empleado> getEmpleadoByGroupId(

**long** groupId, **int** start, **int** end)

**throws** SystemException {

**return** empleadoPersistence.findByGroupId(groupId, start, end);

}

**public** **int** getEmpleadoCountByGroupId(**long** groupId) **throws** SystemException {

**return** empleadoPersistence.countByGroupId(groupId);

}

**public** Empleado updateEmpleado(

**long** empleadoId, **long** userId, **long** groupId, String nombre, String apellido1, String apellido2, String dni,

**long** id\_categoria, **long** id\_cargo, **long** id\_servicio, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

Empleado empleado = EmpleadoLocalServiceUtil.*fetchEmpleado*(empleadoId);

empleado.setModifiedDate(serviceContext.getModifiedDate(now));

empleado.setNombre(nombre);

empleado.setApellido1(apellido1);

empleado.setApellido2(apellido2);

empleado.setDni(dni);

empleado.setId\_cargo(id\_cargo);

empleado.setId\_categoria(id\_categoria);

empleado.setId\_servicio(id\_servicio);

**super**.updateEmpleado(empleado);

**return** empleado;

} }

Y sus imports de librerías :

**import** java.util.Date;

**import** java.util.List;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.model.User;

**import** com.liferay.portal.service.ServiceContext;

**import** es.formacion.model.Empleado;

**import** es.formacion.service.EmpleadoLocalServiceUtil;

**import** es.formacion.service.base.EmpleadoLocalServiceBaseImpl;

* CargoLocalServiceImpl

**public** Cargo addCargo(

**long** userId, **long** groupId, String desc\_cargo, String movil, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

**long** id\_cargo = counterLocalService.increment(Cargo.**class**.getName());

Cargo cargo = cargoPersistence.create(id\_cargo);

cargo.setDesc\_cargo(desc\_cargo);

cargo.setMovil(movil);

cargo.setGroupId(groupId);

cargo.setCompanyId(user.getCompanyId());

cargo.setUserId(user.getUserId());

cargo.setCreateDate(serviceContext.getCreateDate(now));

cargo.setModifiedDate(serviceContext.getModifiedDate(now));

**super**.addCargo(cargo);

**return** cargo;

}

**public** Cargo deleteCargo(Cargo cargo) **throws** SystemException {

**return** cargoPersistence.remove(cargo);

}

**public** Cargo deletecargo(**long** cargoId)

**throws** PortalException, SystemException {

Cargo cargo = cargoPersistence.findByPrimaryKey(cargoId);

**return** deleteCargo(cargo);

}

**public** Cargo getcargo(**long** cargoId)

**throws** SystemException, PortalException {

**return** cargoPersistence.findByPrimaryKey(cargoId);

}

**public** List<Cargo> getCargosByGroupId(**long** groupId)

**throws** SystemException {

**return** cargoPersistence.findByGroupId(groupId);

}

**public** List<Cargo> getCargoByGroupId(

**long** groupId, **int** start, **int** end)

**throws** SystemException {

**return** cargoPersistence.findByGroupId(groupId, start, end);

}

**public** **int** getCargoCountByGroupId(**long** groupId) **throws** SystemException {

**return** cargoPersistence.countByGroupId(groupId);

}

**public** Cargo updateCargo(

**long** cargoId, **long** userId, **long** groupId, String desc\_cargo, String movil, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

Cargo cargo = CargoLocalServiceUtil.*fetchCargo*(cargoId);

cargo.setModifiedDate(serviceContext.getModifiedDate(now));

cargo.setDesc\_cargo(desc\_cargo);

cargo.setMovil(movil);

**super**.updateCargo(cargo);

**return** cargo;

}

Y sus imports de librerías:

**import** java.util.Date;

**import** java.util.List;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.model.User;

**import** com.liferay.portal.service.ServiceContext;

**import** es.formacion.model.Cargo;

**import** es.formacion.model.Cargo;

**import** es.formacion.service.CargoLocalServiceUtil;

**import** es.formacion.service.base.CargoLocalServiceBaseImpl;

* CategoriaLocalServiceImpl

**public** Categoria addCategoria(

**long** userId, **long** groupId, String desc\_categoria, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

**long** id\_categoria = counterLocalService.increment(Categoria.**class**.getName());

Categoria categoria = categoriaPersistence.create(id\_categoria);

categoria.setDesc\_categoria(desc\_categoria);

categoria.setGroupId(groupId);

categoria.setCompanyId(user.getCompanyId());

categoria.setUserId(user.getUserId());

categoria.setCreateDate(serviceContext.getCreateDate(now));

categoria.setModifiedDate(serviceContext.getModifiedDate(now));

**super**.addCategoria(categoria);

**return** categoria;

}

**public** Categoria deleteCategoria(Categoria categoria) **throws** SystemException {

**return** categoriaPersistence.remove(categoria);

}

**public** Categoria deletecategoria(**long** categoriaId)

**throws** PortalException, SystemException {

Categoria categoria = categoriaPersistence.findByPrimaryKey(categoriaId);

**return** deleteCategoria(categoria);

}

**public** Categoria getcategoria(**long** categoriaId)

**throws** SystemException, PortalException {

**return** categoriaPersistence.findByPrimaryKey(categoriaId);

}

**public** List<Categoria> getCategoriasByGroupId(**long** groupId)

**throws** SystemException {

**return** categoriaPersistence.findByGroupId(groupId);

}

**public** List<Categoria> getCategoriaByGroupId(

**long** groupId, **int** start, **int** end)

**throws** SystemException {

**return** categoriaPersistence.findByGroupId(groupId, start, end);

}

**public** **int** getCategoriaCountByGroupId(**long** groupId) **throws** SystemException {

**return** categoriaPersistence.countByGroupId(groupId);

}

**public** Categoria updateCategoria(

**long** categoriaId, **long** userId, **long** groupId, String desc\_categoria, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

Categoria categoria = CategoriaLocalServiceUtil.*fetchCategoria*(categoriaId);

categoria.setModifiedDate(serviceContext.getModifiedDate(now));

categoria.setDesc\_categoria(desc\_categoria);

**super**.updateCategoria(categoria);

**return** categoria;

}

Y sus imports de librerías:

**import** java.util.Date;

**import** java.util.List;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.model.User;

**import** com.liferay.portal.service.ServiceContext;

**import** es.formacion.model.Categoria;

**import** es.formacion.model.Categoria;

**import** es.formacion.service.CategoriaLocalServiceUtil;

**import** es.formacion.service.base.CategoriaLocalServiceBaseImpl;

* ServicioLocalServiceImpl

**public** Servicio addServicio(

**long** userId, **long** groupId, String nombre, String telefono, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

**long** id\_servicio = counterLocalService.increment(Servicio.**class**.getName());

Servicio servicio = servicioPersistence.create(id\_servicio);

servicio.setNombre(nombre);

servicio.setTelefono(telefono);

servicio.setGroupId(groupId);

servicio.setCompanyId(user.getCompanyId());

servicio.setUserId(user.getUserId());

servicio.setCreateDate(serviceContext.getCreateDate(now));

servicio.setModifiedDate(serviceContext.getModifiedDate(now));

**super**.addServicio(servicio);

**return** servicio;

}

**public** Servicio deleteServicio(Servicio servicio) **throws** SystemException {

**return** servicioPersistence.remove(servicio);

}

**public** Servicio deleteservicio(**long** servicioId)

**throws** PortalException, SystemException {

Servicio servicio = servicioPersistence.findByPrimaryKey(servicioId);

**return** deleteServicio(servicio);

}

**public** Servicio getservicio(**long** servicioId)

**throws** SystemException, PortalException {

**return** servicioPersistence.findByPrimaryKey(servicioId);

}

**public** List<Servicio> getServiciosByGroupId(**long** groupId)

**throws** SystemException {

**return** servicioPersistence.findByGroupId(groupId);

}

**public** List<Servicio> getServicioByGroupId(

**long** groupId, **int** start, **int** end)

**throws** SystemException {

**return** servicioPersistence.findByGroupId(groupId, start, end);

}

**public** **int** getServicioCountByGroupId(**long** groupId) **throws** SystemException {

**return** servicioPersistence.countByGroupId(groupId);

}

**public** Servicio updateServicio(

**long** servicioId, **long** userId, **long** groupId, String nombre, String telefono, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

Servicio servicio = ServicioLocalServiceUtil.*fetchServicio*(servicioId);

servicio.setModifiedDate(serviceContext.getModifiedDate(now));

servicio.setNombre(nombre);

servicio.setTelefono(telefono);

**super**.updateServicio(servicio);

**return** servicio;

}

Y sus imports de librerías:

**import** java.util.Date;

**import** java.util.List;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.model.User;

**import** com.liferay.portal.service.ServiceContext;

**import** es.formacion.model.Servicio;

**import** es.formacion.model.Servicio;

**import** es.formacion.service.ServicioLocalServiceUtil;

**import** es.formacion.service.base.ServicioLocalServiceBaseImpl;

Una vez ya lo tenemos listo, volvemos a pulsar el botón de “Build Services” para que nos autogenere las clases \*LocalServiceUtil en las que tendremos los métodos indicados y serán donde invoquemos. También generará el services.properties.

### Llamando a nuestros servicios locales

Ahora, ya podemos en nuestras clases del portlet llamar a éstos servicios locales. Para ello recordar que debemos invocar los métodos de \*LocalServiceUtil. Por tanto modificaremos las clases siguientes:

* ListadoEmpleados

**public** **void** addEmpleado(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateEmpleado(request);

sendRedirect(request, response);

}

**public** **void** deleteEmpleado(ActionRequest request, ActionResponse response)

**throws** Exception {

**long** id\_empleado = ParamUtil.*getLong*(request, "id\_empleado");

EmpleadoLocalServiceUtil.*deleteEmpleado*(id\_empleado);

sendRedirect(request, response);

}

**public** **void** updateEmpleado(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateEmpleado(request);

sendRedirect(request, response);

}

**private** Empleado \_updateEmpleado(ActionRequest request)

**throws** PortalException, SystemException {

**long** id\_empleado = ParamUtil.*getLong*(request, "id\_empleado");

String nombre = ParamUtil.*getString*(request, "nombre");

String apellido1 = ParamUtil.*getString*(request, "apellido1");

String apellido2 = ParamUtil.*getString*(request, "apellido2");

String dni = ParamUtil.*getString*(request, "dni");

**long** categoriaId = ParamUtil.*getLong*(request, "id\_categoria");

**long** cargoId = ParamUtil.*getLong*(request, "id\_cargo");

**long** servicioId = ParamUtil.*getLong*(request, "id\_servicio");

ServiceContext serviceContext = ServiceContextFactory.*getInstance*(

Empleado.**class**.getName(), request);

Empleado empleado = **null**;

**if** (id\_empleado <= 0) {

empleado = EmpleadoLocalServiceUtil.*addEmpleado*(serviceContext.getUserId(), serviceContext.getScopeGroupId(),

nombre, apellido1, apellido2, dni, categoriaId, cargoId, servicioId, serviceContext);

}

**else** {

empleado = EmpleadoLocalServiceUtil.*getEmpleado*(id\_empleado);

empleado = EmpleadoLocalServiceUtil.*updateEmpleado*(id\_empleado, serviceContext.getUserId(),

serviceContext.getScopeGroupId(), nombre,

apellido1, apellido2, dni, categoriaId, cargoId, servicioId, serviceContext);

}

**return** empleado;

}

**private** **static** Log *\_log* = LogFactoryUtil.*getLog*(ListadoEmpleados.**class**);

Y sus imports:

**import** javax.portlet.ActionRequest;

**import** javax.portlet.ActionResponse;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.kernel.log.Log;

**import** com.liferay.portal.kernel.log.LogFactoryUtil;

**import** com.liferay.portal.kernel.util.ParamUtil;

**import** com.liferay.portal.service.ServiceContext;

**import** com.liferay.portal.service.ServiceContextFactory;

**import** com.liferay.util.bridges.mvc.MVCPortlet;

**import** es.formacion.model.Empleado;

**import** es.formacion.service.EmpleadoLocalServiceUtil;

* ListadoCargos

**public** **void** addCargo(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateCargo(request);

sendRedirect(request, response);

}

**public** **void** deleteCargo(ActionRequest request, ActionResponse response)

**throws** Exception {

**long** id\_cargo = ParamUtil.*getLong*(request, "id\_cargo");

CargoLocalServiceUtil.*deleteCargo*(id\_cargo);

sendRedirect(request, response);

}

**public** **void** updateCargo(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateCargo(request);

sendRedirect(request, response);

}

**private** Cargo \_updateCargo(ActionRequest request)

**throws** PortalException, SystemException {

**long** id\_cargo = ParamUtil.*getLong*(request, "id\_cargo");

String desc\_cargo = ParamUtil.*getString*(request, "desc\_cargo");

String movil = ParamUtil.*getString*(request, "movil");

ServiceContext serviceContext = ServiceContextFactory.*getInstance*(

Cargo.**class**.getName(), request);

Cargo cargo = **null**;

**if** (id\_cargo <= 0) {

cargo = CargoLocalServiceUtil.*addCargo*(serviceContext.getUserId(), serviceContext.getScopeGroupId(),

desc\_cargo,movil, serviceContext);

}

**else** {

cargo = CargoLocalServiceUtil.*getCargo*(id\_cargo);

cargo = CargoLocalServiceUtil.*updateCargo*(id\_cargo, serviceContext.getUserId(), serviceContext.getScopeGroupId(),

desc\_cargo, movil, serviceContext);

}

**return** cargo;

}

**private** **static** Log *\_log* = LogFactoryUtil.*getLog*(ListadoCargos.**class**);

Y sus imports

**import** javax.portlet.ActionRequest;

**import** javax.portlet.ActionResponse;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.kernel.log.Log;

**import** com.liferay.portal.kernel.log.LogFactoryUtil;

**import** com.liferay.portal.kernel.util.ParamUtil;

**import** com.liferay.portal.service.ServiceContext;

**import** com.liferay.portal.service.ServiceContextFactory;

**import** com.liferay.util.bridges.mvc.MVCPortlet;

**import** es.formacion.model.Cargo;

**import** es.formacion.service.CargoLocalServiceUtil;

* ListadoCategorias

**public** **void** addCategoria(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateCategoria(request);

sendRedirect(request, response);

}

**public** **void** deleteCategoria(ActionRequest request, ActionResponse response)

**throws** Exception {

**long** id\_categoria = ParamUtil.*getLong*(request, "id\_categoria");

CategoriaLocalServiceUtil.*deleteCategoria*(id\_categoria);

sendRedirect(request, response);

}

**public** **void** updateCategoria(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateCategoria(request);

sendRedirect(request, response);

}

**private** Categoria \_updateCategoria(ActionRequest request)

**throws** PortalException, SystemException {

**long** id\_categoria = ParamUtil.*getLong*(request, "id\_categoria");

String desc\_categoria = ParamUtil.*getString*(request, "desc\_categoria");

ServiceContext serviceContext = ServiceContextFactory.*getInstance*(

Categoria.**class**.getName(), request);

Categoria categoria = **null**;

**if** (id\_categoria <= 0) {

categoria = CategoriaLocalServiceUtil.*addCategoria*(serviceContext.getUserId(), serviceContext.getScopeGroupId(),

desc\_categoria, serviceContext);

}

**else** {

categoria = CategoriaLocalServiceUtil.*getCategoria*(id\_categoria);

categoria = CategoriaLocalServiceUtil.*updateCategoria*(id\_categoria, serviceContext.getUserId(), serviceContext.getScopeGroupId(),

desc\_categoria, serviceContext);

}

**return** categoria;

}

**private** **static** Log *\_log* = LogFactoryUtil.*getLog*(ListadoCategorias.**class**);

Y sus imports

**import** javax.portlet.ActionRequest;

**import** javax.portlet.ActionResponse;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.kernel.log.Log;

**import** com.liferay.portal.kernel.log.LogFactoryUtil;

**import** com.liferay.portal.kernel.util.ParamUtil;

**import** com.liferay.portal.service.ServiceContext;

**import** com.liferay.portal.service.ServiceContextFactory;

**import** com.liferay.util.bridges.mvc.MVCPortlet;

**import** es.formacion.model.Categoria;

**import** es.formacion.service.CategoriaLocalServiceUtil;

* ListadoServicios

**public** **void** addServicio(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateServicio(request);

sendRedirect(request, response);

}

**public** **void** deleteServicio(ActionRequest request, ActionResponse response)

**throws** Exception {

**long** id\_servicio = ParamUtil.*getLong*(request, "id\_servicio");

ServicioLocalServiceUtil.*deleteServicio*(id\_servicio);

sendRedirect(request, response);

}

**public** **void** updateServicio(ActionRequest request, ActionResponse response)

**throws** Exception {

\_updateServicio(request);

sendRedirect(request, response);

}

**private** Servicio \_updateServicio(ActionRequest request)

**throws** PortalException, SystemException {

**long** id\_servicio = ParamUtil.*getLong*(request, "id\_servicio");

String nombre = ParamUtil.*getString*(request, "nombre");

String telefono = ParamUtil.*getString*(request, "telefono");

ServiceContext serviceContext = ServiceContextFactory.*getInstance*(

Servicio.**class**.getName(), request);

Servicio servicio = **null**;

**if** (id\_servicio <= 0) {

servicio = ServicioLocalServiceUtil.*addServicio*(serviceContext.getUserId(), serviceContext.getScopeGroupId(),

nombre, telefono, serviceContext);

}

**else** {

servicio = ServicioLocalServiceUtil.*getServicio*(id\_servicio);

servicio = ServicioLocalServiceUtil.*updateServicio*(id\_servicio, serviceContext.getUserId(), serviceContext.getScopeGroupId(),

nombre, telefono, serviceContext);

}

**return** servicio;

}

**private** **static** Log *\_log* = LogFactoryUtil.*getLog*(ListadoServicios.**class**);

Y sus imports

**import** javax.portlet.ActionRequest;

**import** javax.portlet.ActionResponse;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.kernel.log.Log;

**import** com.liferay.portal.kernel.log.LogFactoryUtil;

**import** com.liferay.portal.kernel.util.ParamUtil;

**import** com.liferay.portal.service.ServiceContext;

**import** com.liferay.portal.service.ServiceContextFactory;

**import** com.liferay.util.bridges.mvc.MVCPortlet;

**import** es.formacion.model.Servicio;

**import** es.formacion.service.ServicioLocalServiceUtil;

### Interfaz

Una vez ya tenemos el código bastante avanzado, vamos a empezar a visualizar y llamar a nuestras clases desde JSP.

* Listadoempleados – view.jsp

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/theme"* prefix=*"liferay-theme"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ taglib uri=*"http://liferay.com/tld/aui"* prefix=*"aui"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Empleado"*%>

<%@ page import=*"es.formacion.service.EmpleadoLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Cargo"*%>

<%@ page import=*"es.formacion.service.CargoLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Categoria"*%>

<%@ page import=*"es.formacion.service.CategoriaLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Servicio"*%>

<%@ page import=*"es.formacion.service.ServicioLocalServiceUtil"*%>

<liferay-theme:defineObjects />

<portlet:defineObjects />

This is the <b>Empleados Portlet</b> in View mode.

<%!

com.liferay.portal.kernel.dao.search.SearchContainer searchContainer = **null**;

%>

<liferay-ui:search-container emptyResultsMessage=*"No hay empleados que mostrar"*>

<liferay-ui:search-container-results

results=*"*<%= EmpleadoLocalServiceUtil.getEmpleadoByGroupId(scopeGroupId, searchContainer.getStart(), searchContainer.getEnd()) %>*"*

total=*"*<%= EmpleadoLocalServiceUtil.getEmpleadoCountByGroupId(scopeGroupId) %>*"*

/>

<liferay-ui:search-container-row

className=*"es.formacion.model.Empleado"*

keyProperty=*"id\_empleado"*

modelVar=*"empleado"* escapedModel=*"*<%= **true** %>*"*

>

<liferay-ui:search-container-column-text

name=*"nombre"*

value=*"*<%= empleado.getNombre() %>*"*

/>

<liferay-ui:search-container-column-text

name=*"apellido1"*

value=*"*<%= empleado.getApellido1() %>*"*

/>

<liferay-ui:search-container-column-text

name=*"apellido2"*

value=*"*<%= empleado.getApellido2() %>*"*

/>

<liferay-ui:search-container-column-text

name=*"dni"*

value=*"*<%= empleado.getDni() %>*"*

/>

<%

String CategoriaNombre = "";

**try** {

CategoriaNombre = CategoriaLocalServiceUtil.getCategoria(empleado.getId\_categoria()).getDesc\_categoria();

} **catch** (Exception e) {

}

String CargoNombre = "";

**try** {

CargoNombre = CargoLocalServiceUtil.getCargo(empleado.getId\_cargo()).getDesc\_cargo();

} **catch** (Exception e) {

}

String ServicioNombre = "";

**try** {

ServicioNombre = ServicioLocalServiceUtil.getServicio(empleado.getId\_servicio()).getNombre();

} **catch** (Exception e) {

}

%>

<liferay-ui:search-container-column-text

name=*"categoria"*

value=*"*<%= CategoriaNombre %>*"*

/>

<liferay-ui:search-container-column-text

name=*"cargo"*

value=*"*<%= CargoNombre %>*"*

/>

<liferay-ui:search-container-column-text

name=*"servicio"*

value=*"*<%= ServicioNombre %>*"*

/>

</liferay-ui:search-container-row>

<liferay-ui:search-iterator />

</liferay-ui:search-container>

* Listadocategorias – view.jsp

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/theme"* prefix=*"liferay-theme"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ taglib uri=*"http://liferay.com/tld/aui"* prefix=*"aui"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Categoria"*%>

<%@ page import=*"es.formacion.service.CategoriaLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Servicio"*%>

<%@ page import=*"es.formacion.service.ServicioLocalServiceUtil"*%>

<liferay-theme:defineObjects />

<portlet:defineObjects />

This is the <b>Categorias Portlet</b> in View mode.

<%!

com.liferay.portal.kernel.dao.search.SearchContainer searchContainer = **null**;

%>

<liferay-ui:search-container emptyResultsMessage=*"No hay categorias que mostrar"*>

<liferay-ui:search-container-results

results=*"*<%= CategoriaLocalServiceUtil.getCategoriaByGroupId(scopeGroupId, searchContainer.getStart(), searchContainer.getEnd()) %>*"*

total=*"*<%= CategoriaLocalServiceUtil.getCategoriaCountByGroupId(scopeGroupId) %>*"*

/>

<liferay-ui:search-container-row

className=*"es.formacion.model.Categoria"*

keyProperty=*"id\_categoria"*

modelVar=*"categoria"* escapedModel=*"*<%= **true** %>*"*

>

<liferay-ui:search-container-column-text

name=*"desc\_categoria"*

value=*"*<%= categoria.getDesc\_categoria() %>*"*

/>

</liferay-ui:search-container-row>

<liferay-ui:search-iterator />

</liferay-ui:search-container>

* Listadocargos – view.jsp

<%@page import=*"es.formacion.service.CargoLocalServiceUtil"*%>

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/theme"* prefix=*"liferay-theme"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ taglib uri=*"http://liferay.com/tld/aui"* prefix=*"aui"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Categoria"*%>

<%@ page import=*"es.formacion.service.CategoriaLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Servicio"*%>

<%@ page import=*"es.formacion.service.ServicioLocalServiceUtil"*%>

<liferay-theme:defineObjects />

<portlet:defineObjects />

This is the <b>Cargos Portlet</b> in View mode.

<%!

com.liferay.portal.kernel.dao.search.SearchContainer searchContainer = **null**;

%>

<liferay-ui:search-container emptyResultsMessage=*"No hay cargos que mostrar"*>

<liferay-ui:search-container-results

results=*"*<%= CargoLocalServiceUtil.getCargoByGroupId(scopeGroupId, searchContainer.getStart(), searchContainer.getEnd()) %>*"*

total=*"*<%= CargoLocalServiceUtil.getCargoCountByGroupId(scopeGroupId) %>*"*

/>

<liferay-ui:search-container-row

className=*"es.formacion.model.Cargo"*

keyProperty=*"id\_cargo"*

modelVar=*"cargo"* escapedModel=*"*<%= **true** %>*"*

>

<liferay-ui:search-container-column-text

name=*"desc\_cargo"*

value=*"*<%= cargo.getDesc\_cargo() %>*"*

/>

<liferay-ui:search-container-column-text

name=*"movil"*

value=*"*<%= cargo.getMovil() %>*"*

/>

</liferay-ui:search-container-row>

<liferay-ui:search-iterator />

</liferay-ui:search-container>

* Listadoservicios – view.jsp

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/theme"* prefix=*"liferay-theme"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ taglib uri=*"http://liferay.com/tld/aui"* prefix=*"aui"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Categoria"*%>

<%@ page import=*"es.formacion.service.CategoriaLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Servicio"*%>

<%@ page import=*"es.formacion.service.ServicioLocalServiceUtil"*%>

<liferay-theme:defineObjects />

<portlet:defineObjects />

This is the <b>Servicios Portlet</b> in View mode.

<%!

com.liferay.portal.kernel.dao.search.SearchContainer searchContainer = **null**;

%>

<liferay-ui:search-container emptyResultsMessage=*"No hay servicios que mostrar"*>

<liferay-ui:search-container-results

results=*"*<%= ServicioLocalServiceUtil.getServicioByGroupId(scopeGroupId, searchContainer.getStart(), searchContainer.getEnd()) %>*"*

total=*"*<%= ServicioLocalServiceUtil.getServicioCountByGroupId(scopeGroupId) %>*"*

/>

<liferay-ui:search-container-row

className=*"es.formacion.model.Servicio"*

keyProperty=*"id\_servicio"*

modelVar=*"servicio"* escapedModel=*"*<%= **true** %>*"*

>

<liferay-ui:search-container-column-text

name=*"nombre"*

value=*"*<%= servicio.getNombre() %>*"*

/>

<liferay-ui:search-container-column-text

name=*"telefono"*

value=*"*<%= servicio.getTelefono() %>*"*

/>

</liferay-ui:search-container-row>

<liferay-ui:search-iterator />

</liferay-ui:search-container>

Previamente a ejecutar y ver el resultado, debemos configurar una dependencia con ciertas librerías en el fichero “liferay-plugin-package.properties”

portal-dependency-jars=\

jstl-api.jar,\

jstl-impl.jar

portal-dependency-tlds=\

liferay-theme.tld,\

liferay-portlet.tld,\

c.tld

Ya podemos visualizar nuestros portlets en Liferay. Observamos que aún no contienen información, sin embargo el Service Builder si nos ha autogenerado las tablas en base de datos “formacionservices\_empleados”, “formacionservices\_cargos”, “formacionservices\_categorias”, “formacionservices\_servicios”.

Ahora vamos a ampliar la funcionalidad con nuevos JSPs de tal forma que podamos hacer inserts, deletes y updates de los datos de esas tablas.

### Ampliando funcionalidad

Añadiremos nuevos JSPs a cada uno de los portlets:

* listadoempleados
  + edit\_empleado.jsp

<%@ include file=*"/html/init.jsp"* %>

<%

Empleado empleado = **null**;

**long** empleadoId = ParamUtil.getLong(request, "empleadoId");

**if** (empleadoId > 0) {

empleado = EmpleadoLocalServiceUtil.getEmpleado(empleadoId);

}

List<Categoria> categorias = CategoriaLocalServiceUtil.getCategoriasByGroupId(scopeGroupId);

List<Cargo> cargos = CargoLocalServiceUtil.getCargosByGroupId(scopeGroupId);

List<Servicio> servicios = ServicioLocalServiceUtil.getServiciosByGroupId(scopeGroupId);

String redirect = ParamUtil.getString(request, "redirect");

%>

<aui:model-context bean=*"*<%= empleado %>*"* model=*"*<%= Empleado.**class** %>*"* />

<portlet:renderURL var=*"viewEmpleadoURL"* />

<portlet:actionURL name=*'*<%= empleado == **null** ? "addEmpleado" : "updateEmpleado" %>*'* var=*"editEmpleadoURL"* windowState=*"normal"* />

<liferay-ui:header

backURL=*"*<%= viewEmpleadoURL %>*"*

title=*'*<%= (empleado != **null**) ? empleado.getNombre() : "Nuevo Empleado" %>*'*

/>

<aui:form action=*"*<%= editEmpleadoURL %>*"* method=*"POST"* name=*"fm"*>

<aui:fieldset>

<aui:input name=*"redirect"* type=*"hidden"* value=*"*<%= redirect %>*"* />

<aui:input name=*"id\_empleado"* type=*"hidden"* value=*'*<%= empleado == **null** ? "" : empleado.getId\_empleado() %>*'*/>

<aui:input name=*"nombre"* />

<aui:input name=*"apellido1"* />

<aui:input name=*"apellido2"* />

<aui:input name=*"dni"* />

<aui:select label=*"Categoria"* name=*"id\_categoria"* showEmptyOption=*"*<%= **true** %>*"*>

<%

**for** (Categoria categoria : categorias) {

%>

<aui:option selected=*"*<%= empleado != **null** && categoria.getId\_categoria() == empleado.getId\_categoria() %>*"* value=*"*<%= categoria.getId\_categoria() %>*"*><%= categoria.getDesc\_categoria() %></aui:option>

<%

}

%>

</aui:select>

<aui:select label=*"Cargo"* name=*"id\_cargo"* showEmptyOption=*"*<%= **true** %>*"*>

<%

**for** (Cargo cargo : cargos) {

%>

<aui:option selected=*"*<%= empleado != **null** && cargo.getId\_cargo() == empleado.getId\_cargo() %>*"* value=*"*<%= cargo.getId\_cargo() %>*"*><%= cargo.getDesc\_cargo() %></aui:option>

<%

}

%>

</aui:select>

<aui:select label=*"Servicio"* name=*"id\_servicio"* showEmptyOption=*"*<%= **true** %>*"*>

<%

**for** (Servicio servicio : servicios) {

%>

<aui:option selected=*"*<%= empleado != **null** && servicio.getId\_servicio() == empleado.getId\_servicio() %>*"* value=*"*<%= servicio.getId\_servicio() %>*"*><%= servicio.getNombre() %></aui:option>

<%

}

%>

</aui:select>

</aui:fieldset>

<aui:button-row>

<aui:button type=*"submit"* />

<aui:button onClick=*"*<%= viewEmpleadoURL %>*"* type=*"cancel"* />

</aui:button-row>

</aui:form>

* + empleado\_actions.jsp

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ page import=*"com.liferay.portal.kernel.dao.search.ResultRow"* %>

<%@ page import=*"com.liferay.portal.kernel.util.WebKeys"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Empleado"*%>

<portlet:defineObjects />

<%

ResultRow row = (ResultRow) request

.getAttribute(WebKeys.SEARCH\_CONTAINER\_RESULT\_ROW);

Empleado empleado = (Empleado) row.getObject();

**long** groupId = empleado.getGroupId();

String name = Empleado.**class**.getName();

**long** empleadoId = empleado.getId\_empleado();

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<liferay-ui:icon-menu>

<portlet:renderURL var=*"editURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadoempleados/edit\_empleado.jsp"* />

<portlet:param name=*"id\_empleado"* value=*"*<%= String.valueOf(empleadoId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<liferay-ui:icon image=*"edit"* url=*"*<%= editURL.toString() %>*"* />

<portlet:actionURL name=*"deleteEmpleado"* var=*"deleteURL"*>

<portlet:param name=*"id\_empleado"* value=*"*<%= String.valueOf(empleadoId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:actionURL>

<liferay-ui:icon-delete url=*"*<%= deleteURL.toString() %>*"* />

</liferay-ui:icon-menu>

, y ya podemos añadir ambos JSP al principal de listadoempleados – view.jsp. Exactamente añadiremos

<%

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<aui:button-row>

<portlet:renderURL var=*"addFacultadURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadoempleados/edit\_empleado.jsp"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<aui:button onClick=*"*<%= addFacultadURL.toString() %>*"* value=*"Añadir Empleado"* />

</aui:button-row>

Antes de la definición de “searchContainer” , y

<liferay-ui:search-container-column-jsp

align=*"right"*

path=*"/html/listadoempleados/empleado\_actions.jsp"*

/>

Antes de terminar “</liferay-ui:search-container-row>”.

* listadocategorias
  + edit\_categoria.jsp

<%@ include file=*"/html/init.jsp"* %>

<%

Categoria categoria = **null**;

**long** categoriaId = ParamUtil.getLong(request, "id\_categoria");

**if** (categoriaId > 0) {

categoria = CategoriaLocalServiceUtil.getCategoria(categoriaId);

}

String redirect = ParamUtil.getString(request, "redirect");

%>

<aui:model-context bean=*"*<%= categoria %>*"* model=*"*<%= Categoria.**class** %>*"* />

<portlet:renderURL var=*"viewCategoriaURL"* />

<portlet:actionURL name=*'*<%= categoria == **null** ? "addCategoria" : "updateCategoria" %>*'* var=*"editCategoriaURL"* windowState=*"normal"* />

<liferay-ui:header

backURL=*"*<%= viewCategoriaURL %>*"*

title=*'*<%= (categoria != **null**) ? categoria.getDesc\_categoria() : "Nuevo Categoria" %>*'*

/>

<aui:form action=*"*<%= editCategoriaURL %>*"* method=*"POST"* name=*"fm"*>

<aui:fieldset>

<aui:input name=*"redirect"* type=*"hidden"* value=*"*<%= redirect %>*"* />

<aui:input name=*"id\_categoria"* type=*"hidden"* value=*'*<%= categoria == **null** ? "" : categoria.getId\_categoria() %>*'*/>

<aui:input name=*"desc\_categoria"* />

</aui:fieldset>

<aui:button-row>

<aui:button type=*"submit"* />

<aui:button onClick=*"*<%= viewCategoriaURL %>*"* type=*"cancel"* />

</aui:button-row>

</aui:form>

* + categoria\_actions.jsp

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ page import=*"com.liferay.portal.kernel.dao.search.ResultRow"* %>

<%@ page import=*"com.liferay.portal.kernel.util.WebKeys"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Categoria"*%>

<portlet:defineObjects />

<%

ResultRow row = (ResultRow) request

.getAttribute(WebKeys.SEARCH\_CONTAINER\_RESULT\_ROW);

Categoria categoria = (Categoria) row.getObject();

**long** groupId = categoria.getGroupId();

String name = Categoria.**class**.getName();

**long** categoriaId = categoria.getId\_categoria();

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<liferay-ui:icon-menu>

<portlet:renderURL var=*"editURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadocategorias/edit\_categoria.jsp"* />

<portlet:param name=*"id\_categoria"* value=*"*<%= String.valueOf(categoriaId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<liferay-ui:icon image=*"edit"* url=*"*<%= editURL.toString() %>*"* />

<portlet:actionURL name=*"deleteCategoria"* var=*"deleteURL"*>

<portlet:param name=*"id\_categoria"* value=*"*<%= String.valueOf(categoriaId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:actionURL>

<liferay-ui:icon-delete url=*"*<%= deleteURL.toString() %>*"* />

</liferay-ui:icon-menu>

, y ya podemos añadir ambos JSP al principal de listadocategorias – view.jsp. Exactamente añadiremos

<%

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<aui:button-row>

<portlet:renderURL var=*"addCategoriaURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadocategorias/edit\_categoria.jsp"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<aui:button onClick=*"*<%= addCategoriaURL.toString() %>*"* value=*"Añadir Categoria"* />

</aui:button-row>

Antes de la definición de “searchContainer” , y

<liferay-ui:search-container-column-jsp

align=*"right"*

path=*"/html/listadocategorias/categoria\_actions.jsp"*

/>

Antes de terminar “</liferay-ui:search-container-row>”.

* listadocargos
  + edit\_cargo.jsp

<%@ include file=*"/html/init.jsp"* %>

<%

Cargo cargo = **null**;

**long** cargoId = ParamUtil.getLong(request, "id\_cargo");

**if** (cargoId > 0) {

cargo = CargoLocalServiceUtil.getCargo(cargoId);

}

String redirect = ParamUtil.getString(request, "redirect");

%>

<aui:model-context bean=*"*<%= cargo %>*"* model=*"*<%= Cargo.**class** %>*"* />

<portlet:renderURL var=*"viewCargoURL"* />

<portlet:actionURL name=*'*<%= cargo == **null** ? "addCargo" : "updateCargo" %>*'* var=*"editCargoURL"* windowState=*"normal"* />

<liferay-ui:header

backURL=*"*<%= viewCargoURL %>*"*

title=*'*<%= (cargo != **null**) ? cargo.getDesc\_cargo() : "Nuevo Cargo" %>*'*

/>

<aui:form action=*"*<%= editCargoURL %>*"* method=*"POST"* name=*"fm"*>

<aui:fieldset>

<aui:input name=*"redirect"* type=*"hidden"* value=*"*<%= redirect %>*"* />

<aui:input name=*"id\_cargo"* type=*"hidden"* value=*'*<%= cargo == **null** ? "" : cargo.getId\_cargo() %>*'*/>

<aui:input name=*"desc\_cargo"* />

<aui:input name=*"movil"* />

</aui:fieldset>

<aui:button-row>

<aui:button type=*"submit"* />

<aui:button onClick=*"*<%= viewCargoURL %>*"* type=*"cancel"* />

</aui:button-row>

</aui:form>

* + cargo\_actions.jsp

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ page import=*"com.liferay.portal.kernel.dao.search.ResultRow"* %>

<%@ page import=*"com.liferay.portal.kernel.util.WebKeys"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Cargo"*%>

<portlet:defineObjects />

<%

ResultRow row = (ResultRow) request

.getAttribute(WebKeys.SEARCH\_CONTAINER\_RESULT\_ROW);

Cargo cargo = (Cargo) row.getObject();

**long** groupId = cargo.getGroupId();

String name = Cargo.**class**.getName();

**long** cargoId = cargo.getId\_cargo();

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<liferay-ui:icon-menu>

<portlet:renderURL var=*"editURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadocargos/edit\_cargo.jsp"* />

<portlet:param name=*"id\_cargo"* value=*"*<%= String.valueOf(cargoId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<liferay-ui:icon image=*"edit"* url=*"*<%= editURL.toString() %>*"* />

<portlet:actionURL name=*"deleteCargo"* var=*"deleteURL"*>

<portlet:param name=*"id\_cargo"* value=*"*<%= String.valueOf(cargoId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:actionURL>

<liferay-ui:icon-delete url=*"*<%= deleteURL.toString() %>*"* />

</liferay-ui:icon-menu>

, y ya podemos añadir ambos JSP al principal de listadocargos – view.jsp. Exactamente añadiremos

<%

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<aui:button-row>

<portlet:renderURL var=*"addCargoURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadocargos/edit\_cargo.jsp"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<aui:button onClick=*"*<%= addCargoURL.toString() %>*"* value=*"Añadir Cargo"* />

</aui:button-row>

Antes de la definición de “searchContainer” , y

<liferay-ui:search-container-column-jsp

align=*"right"*

path=*"/html/listadocargos/cargo\_actions.jsp"*

/>

Antes de terminar “</liferay-ui:search-container-row>”.

* listadoservicios
  + edit\_servicio.jsp

<%@ include file=*"/html/init.jsp"* %>

<%

Servicio servicio = **null**;

**long** servicioId = ParamUtil.getLong(request, "id\_servicio");

**if** (servicioId > 0) {

servicio = ServicioLocalServiceUtil.getServicio(servicioId);

}

String redirect = ParamUtil.getString(request, "redirect");

%>

<aui:model-context bean=*"*<%= servicio %>*"* model=*"*<%= Servicio.**class** %>*"* />

<portlet:renderURL var=*"viewServicioURL"* />

<portlet:actionURL name=*'*<%= servicio == **null** ? "addServicio" : "updateServicio" %>*'* var=*"editServicioURL"* windowState=*"normal"* />

<liferay-ui:header

backURL=*"*<%= viewServicioURL %>*"*

title=*'*<%= (servicio != **null**) ? servicio.getNombre() : "Nuevo Servicio" %>*'*

/>

<aui:form action=*"*<%= editServicioURL %>*"* method=*"POST"* name=*"fm"*>

<aui:fieldset>

<aui:input name=*"redirect"* type=*"hidden"* value=*"*<%= redirect %>*"* />

<aui:input name=*"id\_servicio"* type=*"hidden"* value=*'*<%= servicio == **null** ? "" : servicio.getId\_servicio() %>*'*/>

<aui:input name=*"desc\_servicio"* />

<aui:input name=*"nombre"* />

<aui:input name=*"telefono"* />

</aui:fieldset>

<aui:button-row>

<aui:button type=*"submit"* />

<aui:button onClick=*"*<%= viewServicioURL %>*"* type=*"cancel"* />

</aui:button-row>

</aui:form>

* + servicio\_actions.jsp

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ page import=*"com.liferay.portal.kernel.dao.search.ResultRow"* %>

<%@ page import=*"com.liferay.portal.kernel.util.WebKeys"* %>

<%@ page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"es.formacion.model.Servicio"*%>

<portlet:defineObjects />

<%

ResultRow row = (ResultRow) request

.getAttribute(WebKeys.SEARCH\_CONTAINER\_RESULT\_ROW);

Servicio servicio = (Servicio) row.getObject();

**long** groupId = servicio.getGroupId();

String name = Servicio.**class**.getName();

**long** servicioId = servicio.getId\_servicio();

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<liferay-ui:icon-menu>

<portlet:renderURL var=*"editURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadoservicios/edit\_servicio.jsp"* />

<portlet:param name=*"id\_servicio"* value=*"*<%= String.valueOf(servicioId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<liferay-ui:icon image=*"edit"* url=*"*<%= editURL.toString() %>*"* />

<portlet:actionURL name=*"deleteServicio"* var=*"deleteURL"*>

<portlet:param name=*"id\_servicio"* value=*"*<%= String.valueOf(servicioId) %>*"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:actionURL>

<liferay-ui:icon-delete url=*"*<%= deleteURL.toString() %>*"* />

</liferay-ui:icon-menu>

, y ya podemos añadir ambos JSP al principal de listadoservicio – view.jsp. Exactamente añadiremos

<%

String redirect = PortalUtil.getCurrentURL(renderRequest);

%>

<aui:button-row>

<portlet:renderURL var=*"addServicioURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadoservicios/edit\_servicio.jsp"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<aui:button onClick=*"*<%= addServicioURL.toString() %>*"* value=*"Añadir Servicio"* />

</aui:button-row>

Antes de la definición de “searchContainer” , y

<liferay-ui:search-container-column-jsp

align=*"right"*

path=*"/html/listadoservicios/servicio\_actions.jsp"*

/>

Antes de terminar “</liferay-ui:search-container-row>”.

, y por último tendremos que apoyarnos en un fichero que pondremos en /HTML que inicialice variables:

* init.jsp

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

<%@ taglib uri=*"http://java.sun.com/portlet\_2\_0"* prefix=*"portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/portlet"* prefix=*"liferay-portlet"* %>

<%@ taglib uri=*"http://liferay.com/tld/theme"* prefix=*"liferay-theme"* %>

<%@ taglib uri=*"http://liferay.com/tld/ui"* prefix=*"liferay-ui"* %>

<%@ taglib uri=*"http://liferay.com/tld/aui"* prefix=*"aui"* %>

<%@ page import=*"com.liferay.portal.kernel.dao.search.ResultRow"* %><%@

page import=*"com.liferay.portal.kernel.util.Constants"* %><%@

page import=*"com.liferay.portal.kernel.util.GetterUtil"* %><%@

page import=*"com.liferay.portal.kernel.util.ParamUtil"* %><%@

page import=*"com.liferay.portal.kernel.util.StringPool"* %><%@

page import=*"com.liferay.portal.kernel.util.WebKeys"* %><%@

page import=*"com.liferay.portal.model.Group"* %><%@

page import=*"com.liferay.portal.theme.ThemeDisplay"* %><%@

page import=*"com.liferay.portal.util.PortalUtil"* %>

<%@ page import=*"java.text.SimpleDateFormat"* %><%@

page import=*"java.util.List"* %>

<%@ page import=*"es.formacion.model.Empleado"*%>

<%@ page import=*"es.formacion.service.EmpleadoLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Cargo"*%>

<%@ page import=*"es.formacion.service.CargoLocalServiceUtil"*%>

<%@ page import=*"es.formacion.model.Categoria"*%>

<%@ page import=*"es.formacion.service.CategoriaLocalServiceUtil"*%>

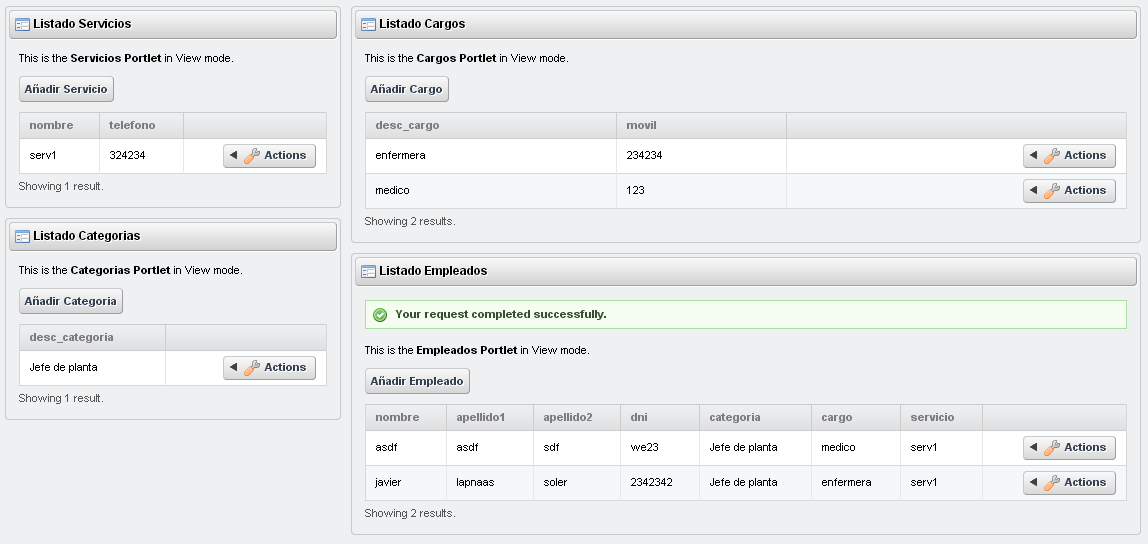
<%@ page import=*"es.formacion.model.Servicio"*%>

<%@ page import=*"es.formacion.service.ServicioLocalServiceUtil"*%>

<liferay-theme:defineObjects />

<portlet:defineObjects />

, por tanto si desplegamos tendremos ya una funcionalidad completa, que deberá parecerse a lo siguiente:



***Ejercicio práctico : añadir un buscador por campos. Referencia:***

[***http://www.liferaysavvy.com/2014/05/liferay-search-form-with-search.html***](http://www.liferaysavvy.com/2014/05/liferay-search-form-with-search.html)

### Acceso a recursos

Una vez tenemos la base de nuestros desarrollos vamos a ver cómo podemos indicar los permisos y qué roles pueden hacer qué acciones en ellos.

Para ello generamos en “docroot/WEB-INF/src” un fichero portal.properties:

resource.actions.configs=resource-actions/default.xml

Y una carpeta: resource-actions con un fichero default.xml que contiene:

<?xml version=*"1.0"*?>

<!DOCTYPE resource-action-mapping PUBLIC "-//Liferay//DTD Resource Action Mapping 6.2.0//EN" "http://www.liferay.com/dtd/liferay-resource-action-mapping\_6\_2\_0.dtd">

<resource-action-mapping>

<portlet-resource>

<portlet-name>listadoempleados</portlet-name>

<permissions>

<supports>

<action-key>ADD\_TO\_PAGE</action-key>

<action-key>CONFIGURATION</action-key>

<action-key>VIEW</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported />

</permissions>

</portlet-resource>

<portlet-resource>

<portlet-name>listadocargos</portlet-name>

<permissions>

<supports>

<action-key>ADD\_TO\_PAGE</action-key>

<action-key>CONFIGURATION</action-key>

<action-key>VIEW</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported />

</permissions>

</portlet-resource>

<portlet-resource>

<portlet-name>listadocategorias</portlet-name>

<permissions>

<supports>

<action-key>ADD\_TO\_PAGE</action-key>

<action-key>CONFIGURATION</action-key>

<action-key>VIEW</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported>

<action-key>CONFIGURATION</action-key>

</guest-unsupported>

</permissions>

</portlet-resource>

<portlet-resource>

<portlet-name>listadoservicios</portlet-name>

<permissions>

<supports>

<action-key>ADD\_TO\_PAGE</action-key>

<action-key>CONFIGURATION</action-key>

<action-key>VIEW</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported />

</permissions>

</portlet-resource>

<model-resource>

<model-name>es.formacion.model.Empleado</model-name>

<portlet-ref>

<portlet-name>listadoempleados</portlet-name>

</portlet-ref>

<permissions>

<supports>

<action-key>VIEW</action-key>

<action-key>ADD\_EMPLEADO</action-key>

<action-key>DELETE\_EMPLEADO</action-key>

<action-key>UPDATE\_EMPLEADO</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported>

<action-key>ADD\_EMPLEADO</action-key>

<action-key>DELETE\_EMPLEADO</action-key>

<action-key>UPDATE\_EMPLEADO</action-key>

</guest-unsupported>

<owner-defaults>

<action-key>DELETE\_EMPLEADO</action-key>

<action-key>UPDATE\_EMPLEADO</action-key>

<action-key>VIEW</action-key>

</owner-defaults>

</permissions>

</model-resource>

<model-resource>

<model-name>es.formacion.model.Cargo</model-name>

<portlet-ref>

<portlet-name>listadocargos</portlet-name>

</portlet-ref>

<permissions>

<supports>

<action-key>VIEW</action-key>

<action-key>ADD\_CARGO</action-key>

<action-key>DELETE\_CARGOS</action-key>

<action-key>UPDATE\_CARGOS</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported>

<action-key>ADD\_CARGO</action-key>

<action-key>DELETE\_CARGOS</action-key>

<action-key>UPDATE\_CARGOS</action-key>

</guest-unsupported>

<owner-defaults>

<action-key>VIEW</action-key>

<action-key>DELETE\_CARGOS</action-key>

<action-key>UPDATE\_CARGOS</action-key>

</owner-defaults>

</permissions>

</model-resource>

<model-resource>

<model-name>es.formacion.model.Servicio</model-name>

<portlet-ref>

<portlet-name>listadoservicios</portlet-name>

</portlet-ref>

<permissions>

<supports>

<action-key>VIEW</action-key>

<action-key>ADD\_SERVICIO</action-key>

<action-key>DELETE\_SERVICIOS</action-key>

<action-key>UPDATE\_SERVICIOS</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported>

<action-key>ADD\_SERVICIO</action-key>

<action-key>DELETE\_SERVICIOS</action-key>

<action-key>UPDATE\_SERVICIOS</action-key>

</guest-unsupported>

<owner-defaults>

<action-key>VIEW</action-key>

<action-key>DELETE\_SERVICIOS</action-key>

<action-key>UPDATE\_SERVICIOS</action-key>

</owner-defaults>

</permissions>

</model-resource>

<model-resource>

<model-name>es.formacion.model.Categoria</model-name>

<portlet-ref>

<portlet-name>listadocategorias</portlet-name>

</portlet-ref>

<permissions>

<supports>

<action-key>VIEW</action-key>

<action-key>ADD\_CATEGORIAS</action-key>

<action-key>DELETE\_CATEGORIAS</action-key>

<action-key>UPDATE\_CATEGORIAS</action-key>

</supports>

<site-member-defaults>

<action-key>VIEW</action-key>

</site-member-defaults>

<guest-defaults>

<action-key>VIEW</action-key>

</guest-defaults>

<guest-unsupported>

<action-key>ADD\_CATEGORIAS</action-key>

<action-key>DELETE\_CATEGORIAS</action-key>

<action-key>UPDATE\_CATEGORIAS</action-key>

</guest-unsupported>

<owner-defaults>

<action-key>VIEW</action-key>

<action-key>DELETE\_CATEGORIAS</action-key>

<action-key>UPDATE\_CATEGORIAS</action-key>

</owner-defaults>

</permissions>

</model-resource>

</resource-action-mapping>

Para ampliar conocimiento sobre los permisos y las diferentes secciones se recomienda la siguiente url:

<https://www.liferay.com/es/documentation/liferay-portal/6.1/development/-/ai/security-and-permissio-4>

A continuación vamos a añadir todo lo referente a los permisos y accesos a permisos para el listado de Categorías.

***Ejercicio : realizar los mismos pasos para el resto de listados***

Primero vamos a modificar nuestro fichero de servicios “CaterogiaLocalServiceImpl.java”. Exactamente modificamos los métodos siguientes que quedan de la siguiente forma:

**public** Categoria addCategoria(

**long** userId, **long** groupId, String desc\_categoria, ServiceContext serviceContext)

**throws** PortalException, SystemException {

User user = userPersistence.findByPrimaryKey(userId);

Date now = **new** Date();

**long** id\_categoria = counterLocalService.increment(Categoria.**class**.getName());

Categoria categoria = categoriaPersistence.create(id\_categoria);

categoria.setDesc\_categoria(desc\_categoria);

categoria.setGroupId(groupId);

categoria.setCompanyId(user.getCompanyId());

categoria.setUserId(user.getUserId());

categoria.setCreateDate(serviceContext.getCreateDate(now));

categoria.setModifiedDate(serviceContext.getModifiedDate(now));

resourceLocalService.addResources(

categoria.getCompanyId(), categoria.getGroupId(), categoria.getUserId(),

Categoria.**class**.getName(), categoria.getPrimaryKey(), **false**,

**true**, **true**);

**super**.addCategoria(categoria);

**return** categoria;

}

**public** Categoria deleteCategoria(Categoria categoria) **throws** SystemException {

**try** {

resourceLocalService.deleteResource(

categoria.getCompanyId(), Categoria.**class**.getName(),

ResourceConstants.***SCOPE\_INDIVIDUAL***, categoria.getPrimaryKey());

} **catch** (PortalException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** categoriaPersistence.remove(categoria);

}

Recordar relanzar el Service Builder cada vez que modificamos éste fichero. Después una vez que hemos definido que añada en base de datos todo lo referente al registro de los recursos para poder definir los permisos vamos a añadir lo siguiente en los ficheros jsps:

\* View.jsp

Cambiamos el botón de Añadir Categoría por lo siguiente:

<%

**if** (permissionChecker.hasPermission(

scopeGroupId, "es.formacion.model.Categoria",

scopeGroupId, "ADD\_CATEGORIAS")) {

%>

<aui:button-row>

<portlet:renderURL var=*"addCategoriaURL"*>

<portlet:param name=*"mvcPath"* value=*"/html/listadocategorias/edit\_categoria.jsp"* />

<portlet:param name=*"redirect"* value=*"*<%= redirect %>*"* />

</portlet:renderURL>

<aui:button onClick=*"*<%= addCategoriaURL.toString() %>*"* value=*"Añadir Categoria"* />

</aui:button-row>

<%

}

%>

,que no es más que un checkeo de permisos previo para saber si mostrar o no el botón.

\* edit\_categoria.jsp

Después de “String redirect … “ añadimos

**if** (categoria != **null**) {

%>

<liferay-security:permissionsURL

modelResource=*"*<%= Categoria.**class**.getName() %>*"*

modelResourceDescription=*"*<%= categoria.getDesc\_categoria() %>*"*

resourcePrimKey=*"*<%= String.valueOf(categoria.getPrimaryKey()) %>*"*

var=*"entryURL"*

/>

<liferay-ui:icon image=*"permissions"* url=*"*<%= entryURL %>*"* />

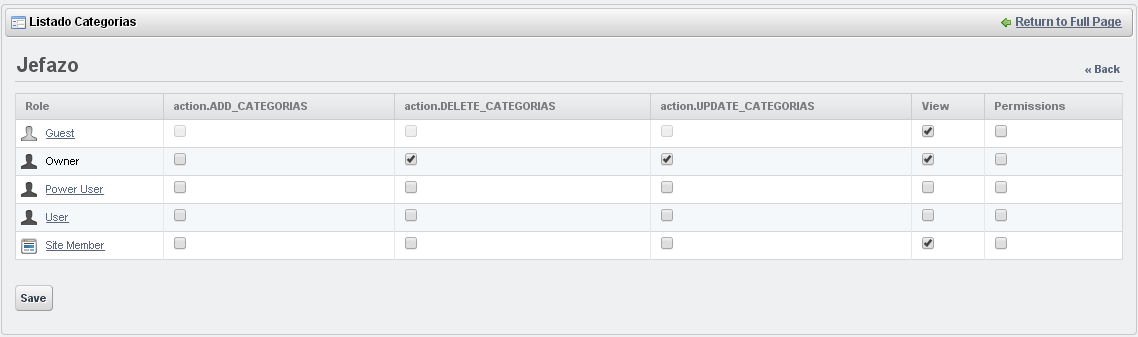
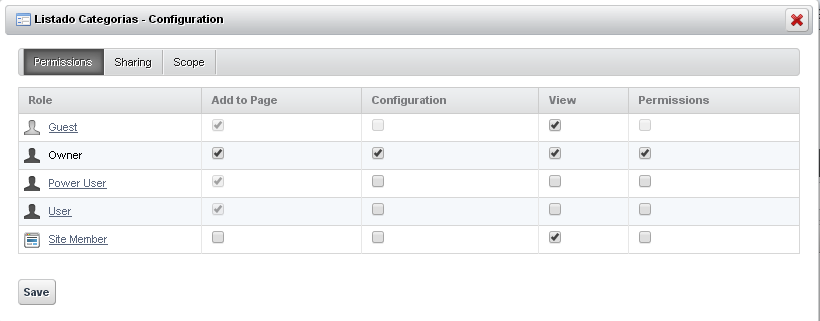
<%

}

%>

Que lo que hacemos es indicar que si estamos editando, poder modificar los permisos de ese recurso, en éste caso una categoría.

Ya podemos desplegar y ver los permisos que nos referimos tanto a nivel portlet como de recursos.



### Scope

Hay que entender que según el ámbito de aplicación de un portlet, los objetos que creemos van a estar vinculados a ese ámbito. Por defecto todo va con scope de Sitio. Vamos a comprobar cómo habilitarlo por ejemplo en el portlet de Listado Categorías y añadiendo dos y cambiando el Scope de uno de ellos a la página actual, podemos tener diferentes listados.

En liferay-portlet.xml, cambiamos la definición de portlet de listado-categorías por lo siguiente:

<portlet>

<portlet-name>**listado-categorias**</portlet-name>

<icon>**/icon.png**</icon>

<instanceable>true</instanceable>

<scopeable>true</scopeable>

<header-portlet-css>**/css/main.css**</header-portlet-css>

<footer-portlet-javascript>

**/js/main.js**

</footer-portlet-javascript>

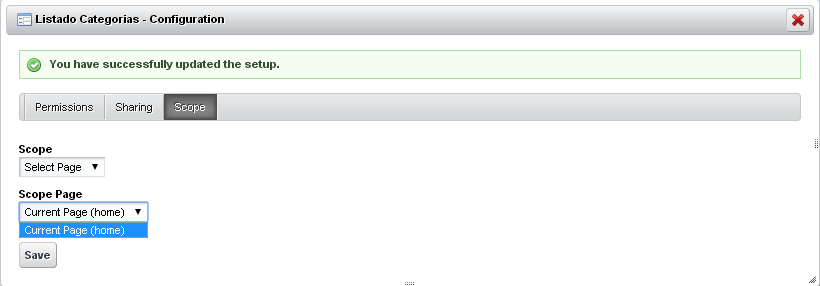
<css-class-wrapper>

listado-categorias-portlet

</css-class-wrapper>

</portlet>

Ahora podemos incluir dos portlets de listado de categorías en nuestra página. En una de ella marcamos en la pestaña de “Configuración” => “Scope” => “Página Actual”



, nos quedaría entonces lo siguiente:



### Servicios Remotos

En Liferay, los Servicios remotos están disponibles para SOAP en

http://localhost:8080/api/axis

y para JSON en

http://localhost:8080/api/jsonws/

La idea es que nosotros hagamos algo parecido y publiquemos nuestros servicios. Primero pulsaremos el botón que hay al lado de “Service Builder” y que se llama “Build WSDD”. Esto nos autogenerará todas las clases necesarias para poder implementar nuestra funcionalidad.

Para implementarla,como sabemos, tenemos que tocar las clases \*ServiceImpl.java. Haremos el ejemplo para ListadoEmpleados.

* EmpleadoServiceImpl

**package** es.formacion.service.impl;

**import** java.util.Date;

**import** java.util.List;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.model.User;

**import** com.liferay.portal.service.ServiceContext;

**import** es.formacion.model.Empleado;

**import** es.formacion.service.EmpleadoLocalServiceUtil;

**import** es.formacion.service.base.EmpleadoServiceBaseImpl;

**import** es.formacion.service.permission.EmpleadoListingPermission;

**import** es.formacion.service.permission.EmpleadoPermission;

**import** es.formacion.util.EmpleadoListingActionKeys;

/\*\*

\* The implementation of the empleado remote service.

\*

\* <p>

\* All custom service methods should be put in this class. Whenever methods are added, rerun ServiceBuilder to copy their definitions into the {@link es.formacion.service.EmpleadoService} interface.

\*

\* <p>

\* This is a remote service. Methods of this service are expected to have security checks based on the propagated JAAS credentials because this service can be accessed remotely.

\* </p>

\*

\* **@author** HCUVA

\* **@see** es.formacion.service.base.EmpleadoServiceBaseImpl

\* **@see** es.formacion.service.EmpleadoServiceUtil

\*/

**public** **class** EmpleadoServiceImpl **extends** EmpleadoServiceBaseImpl {

**public** Empleado addEmpleado(

**long** userId, **long** groupId, String nombre, String apellido1, String apellido2, String dni,

**long** id\_categoria, **long** id\_cargo, **long** id\_servicio, ServiceContext serviceContext)

**throws** PortalException, SystemException {

EmpleadoListingPermission.*check*(

getPermissionChecker(), groupId, EmpleadoListingActionKeys.***ADD\_EMPLEADO***);

**return** EmpleadoLocalServiceUtil.*addEmpleado*(getUserId(), groupId, nombre, apellido1, apellido2, dni,

id\_categoria, id\_cargo, id\_servicio, serviceContext);

}

**public** Empleado deleteempleado(**long** empleadoId)

**throws** PortalException, SystemException {

EmpleadoPermission.*check*(getPermissionChecker(), empleadoId,

EmpleadoListingActionKeys.***DELETE\_EMPLEADO***);

**return** EmpleadoLocalServiceUtil.*deleteEmpleado*(empleadoId);

}

**public** Empleado getempleado(**long** empleadoId)

**throws** SystemException, PortalException {

EmpleadoPermission.*check*(getPermissionChecker(), empleadoId,

EmpleadoListingActionKeys.***VIEW***);

**return** EmpleadoLocalServiceUtil.*getEmpleado*(empleadoId);

}

**public** Empleado updateEmpleado(

**long** empleadoId, **long** userId, **long** groupId, String nombre, String apellido1, String apellido2, String dni,

**long** id\_categoria, **long** id\_cargo, **long** id\_servicio, ServiceContext serviceContext)

**throws** PortalException, SystemException {

EmpleadoPermission.*check*(getPermissionChecker(), empleadoId,

EmpleadoListingActionKeys.***UPDATE\_EMPLEADO***);

**return** EmpleadoLocalServiceUtil.*updateEmpleado*(

empleadoId, userId, groupId, nombre, apellido1, apellido2, dni, id\_categoria, id\_cargo,

id\_servicio, serviceContext);

}

}

En éste caso tendremos que definir clases de apoyo para permisos. Creamos las clases siguientes :

* EmpleadoListingPermission( en paquete es.formacion.service.permission)

**package** es.formacion.service.permission;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.security.auth.PrincipalException;

**import** com.liferay.portal.security.permission.PermissionChecker;

**public** **class** EmpleadoListingPermission {

**public** **static** **void** check(

PermissionChecker permissionChecker, **long** groupId, String actionId)

**throws** PortalException {

**if** (!*contains*(permissionChecker, groupId, actionId)) {

**throw** **new** PrincipalException();

}

}

**public** **static** **boolean** contains(

PermissionChecker permissionChecker, **long** groupId, String actionId) {

**return** permissionChecker.hasPermission(

groupId, ***\_CLASS\_NAME***, groupId, actionId);

}

**private** **static** **final** String ***\_CLASS\_NAME*** = "es.formacion.ListadoEmpleados";

}

* EmpleadoListingActionKeys ( en paquete es.formacion.util )

**package** es.formacion.util;

**import** com.liferay.portal.security.permission.ActionKeys;

**public** **class** EmpleadoListingActionKeys **extends** ActionKeys {

**public** **static** **final** String ***ADD\_EMPLEADO*** = "ADD\_EMPLEADO";

**public** **static** **final** String ***DELETE\_EMPLEADO*** = "DELETE\_EMPLEADO";

**public** **static** **final** String ***UPDATE\_EMPLEADO*** = "UPDATE\_EMPLEADO";

}

* EmpleadoPermission ( en paquete es.formacion.service.permission )

**package** es.formacion.service.permission;

**import** com.liferay.portal.kernel.exception.PortalException;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.security.auth.PrincipalException;

**import** com.liferay.portal.security.permission.PermissionChecker;

**import** es.formacion.model.Empleado;

**import** es.formacion.service.EmpleadoLocalServiceUtil;

**public** **class** EmpleadoPermission {

**public** **static** **void** check(

PermissionChecker permissionChecker, Empleado empleado,

String actionId)

**throws** PortalException {

**if** (!*contains*(permissionChecker, empleado, actionId)) {

**throw** **new** PrincipalException();

}

}

**public** **static** **void** check(

PermissionChecker permissionChecker, **long** id\_empleado, String actionId)

**throws** PortalException, SystemException {

**if** (!*contains*(permissionChecker, id\_empleado, actionId)) {

**throw** **new** PrincipalException();

}

}

**public** **static** **boolean** contains(

PermissionChecker permissionChecker, Empleado empleado,

String actionId) {

**if** (permissionChecker.hasOwnerPermission(

empleado.getCompanyId(), Empleado.**class**.getName(),

empleado.getId\_empleado(), empleado.getUserId(), actionId)) {

**return** **true**;

}

**return** permissionChecker.hasPermission(

empleado.getGroupId(), Empleado.**class**.getName(), empleado.getId\_empleado(),

actionId);

}

**public** **static** **boolean** contains(

PermissionChecker permissionChecker, **long** id\_empleado, String actionId)

**throws** PortalException, SystemException {

Empleado empleado = EmpleadoLocalServiceUtil.*getEmpleado*(id\_empleado);

**return** *contains*(permissionChecker, empleado, actionId);

}

}

Una vez tenemos todo bien compilado, tendremos que llamar a “Build Services” para que nos genere las clases “Utils” para invocar a los servicios remotos. Además deberemos luego también pulsar el botón de al lado “Build WSDD” para que nos genere el fichero de descripción.

Para comprobar que está correcto, en la siguiente url deberíamos ver nuestros servicios:

[*http://localhost:8080/api/jsonws?contextPath=/FormacionService-portlet*](http://localhost:8080/api/jsonws?contextPath=/FormacionService-portlet)

***Ejercicio : Habilitar los servicios remotos para el resto de portlets de listados que tenemos***

### Custom SQL Queries

Desde el Service Builder también nos da la opción más o menos sencilla de definir consultas personalizadas. Vamos a realizar una que filtra los empleados por orden alfabético por nombre y apellido1 además de por el nombre de cargo.

Lo primero es crear la carpeta “custom-sql” en la docroot/WEB-INF/src. En ella introducimos default.xml con el siguiente texto:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<custom-sql>

<sql id=*"es.formacion.service.persistence.EmpleadoFinder.findByEmpleadoNameEmpleadoApellido1CargoNombre"*>

SELECT formacionservice\_empleado.\*

FROM formacionservice\_empleado

INNER JOIN

formacionservice\_cargo ON formacionservice\_empleado.id\_cargo = formacionservice\_cargo.id\_cargo

WHERE

(formacionservice\_empleado.nombre LIKE ?) AND

(formacionservice\_empleado.apellido1 LIKE ?) AND

(formacionservice\_cargo.desc\_cargo LIKE ?)

</sql>

</custom-sql>

Ahora tenemos que crear nuestro “Finder” para que el ServiceBuilder nos genere el código necesario. Para ello creamos la clase EmpleadoFinderImpl en es.formacion.service.persistence con el siguiente código:

**package** es.formacion.service.persistence;

**import** com.liferay.portal.service.persistence.impl.BasePersistenceImpl;

**import** es.formacion.model.Empleado;

**public** **class** EmpleadoFinderImpl **extends** BasePersistenceImpl<Empleado>{

}

Volvemos a lanzar el Service Builder para que nos genere el Finder automáticamente y ya podamos implementarlo:

**package** es.formacion.service.persistence;

**import** java.util.List;

**import** com.liferay.portal.kernel.dao.orm.QueryPos;

**import** com.liferay.portal.kernel.dao.orm.QueryUtil;

**import** com.liferay.portal.kernel.dao.orm.SQLQuery;

**import** com.liferay.portal.kernel.dao.orm.Session;

**import** com.liferay.portal.kernel.exception.SystemException;

**import** com.liferay.portal.service.persistence.impl.BasePersistenceImpl;

**import** com.liferay.util.dao.orm.CustomSQLUtil;

**import** es.formacion.model.Empleado;

**import** es.formacion.model.impl.EmpleadoImpl;

**public** **class** EmpleadoFinderImpl **extends** BasePersistenceImpl<Empleado> **implements** EmpleadoFinder{

**public** List<Empleado> findByEmpleadoNombreEmpleadoApellido1CargoNombre(

String empleadoNombre, String empleadoApellido1, String cargoNombre, **int** begin, **int** end) {

Session session = **null**;

**try** {

session = openSession();

String sql = CustomSQLUtil.*get*(***FIND\_BY\_EMPLEADONOMBRE\_EMPLEADOAPELLIDO1\_CARGONOMBRE***);

SQLQuery q = session.createSQLQuery(sql);

q.setCacheable(**false**);

q.addEntity("Empleado\_Empleado", EmpleadoImpl.**class**);

QueryPos qPos = QueryPos.*getInstance*(q);

qPos.add(empleadoNombre);

qPos.add(empleadoApellido1);

qPos.add(cargoNombre);

**return** (List<Empleado>)QueryUtil.*list*(q, getDialect(), begin, end);

} **catch** (Exception e) {

**try** {

**throw** **new** SystemException(e);

} **catch** (SystemException se) {

se.printStackTrace();

}

} **finally** {

closeSession(session);

}

**return** **null**;

}

**public** **static** **final** String ***FIND\_BY\_EMPLEADONOMBRE\_EMPLEADOAPELLIDO1\_CARGONOMBRE*** = EmpleadoFinder.**class**.getName() + ".findByEmpleadoNameEmpleadoApellido1CargoNombre";

}

Y ya podemos añadirlo en nuestro EmpleadoLocalServiceImpl:

**public** List<Empleado> findByEmpleadoNombreEmpleadoApellido1CargoNombre (

String empleadoNombre, String empleadoApellido1, String cargoNombre, **int** begin, **int** end)

**throws** SystemException {

**return** EmpleadoFinderUtil.findByEmpleadoNombreEmpleadoApellido1CargoNombre(

empleadoNombre, empleadoApellido1, cargoNombre, begin, end);

}

Y volvemos a llamar al ServiceBuilder para que nos genere el método Util correspondiente para poder llamarlo.

Para llamarlo, desde nuestro JSP podríamos hacer perfectamente :

EmpleadoLocalServiceUtil.findByEmpleadoNombreEmpleadoApellido1CargoNombre (...)

***Ejercicio. Realizar otras custom-sql para el resto de listados***