Acciones Liferay

Control de versiones del documento

Versión	Fecha	Autor	Motivo del cambio
1.0	28/05/2015	Antonio Morcillo Martínez	Creación del documento

Tabla de contenido

Acciones *Liferay*

Liferay nos proporciona un mecanismo para la extensión del comportamiento por defecto de la plataforma ante determinados eventos.
En concreto Liferay proporciona tres puntos de extensión:

- *Service:* Este evento se dispara en cada petición procesada por *Liferay* ya sea de página o contenido estático por lo que **se debe ser extremadamente cuidadoso con lo que se hace en este tipo de eventos** ya que impacta enormemente en el tiempo de respuesta del portal
- *Login:* Este evento se genera cuando un usuario se autentica en el sistema mediante los mecanismos estándar de *Liferay*
- *Logout:* Este evento se genera cuando un usuario sale de la plataforma mediante los mecanismos estándar de *Liferay*

Cada uno de estos tres eventos nos ofrece dos posibilidades de añadir comportamiento a su funcionamiento por defecto, *pre y post,* es decir nuestro código se ejecutará antes o después del código de *Liferay* que atiende al evento. Aunque es posible sustituir por completo el comportamiento de *Liferay*, en este documento nos centraremos en cómo complementar el comportamiento de *Liferay*.
Este mecanismo de extensión puede emplearse mediante un *plugin EXT* o bien mediante la mucho más recomendable opción de un *plugin Hook*. A continuación se describen los pasos para la creación de un *Hook* que incorpore acciones.

Creación de un *Hook* con *Maven*

Mediante *maven* es extremadamente sencillo crear un *hook*, ya que disponemos de arquetipos que nos generan la estructura completa de un proyecto con todos los ficheros de configuración necesarios para poder compilar e instalar *Hooks*.
Asumimos que disponemos de un entorno de desarrollo con *Maven* y el *SDK* de *Liferay* para *Maven* correctamente instalado. Para más información:

```
http://www.liferay.com/es/community/wiki/-/wiki/Main/Liferay+Maven+SDK
```

Para crear un proyecto con *Maven* en modo interactivo siguiendo un arquetipo deberemos lanzar el siguiente comando

```
mvn archetype:generate
```

Nos aparecerá una lista con todos los arquetipos disponibles. Como la lista es, potencialmente, muy grande podremos filtrarla limitándola a los arquetipos en los que estamos interesados.
Para ello escribiremos 'liferay' cuando *maven* nos pregunte que escribamos el grupo del arquetipo.
Si todo ha ido bien nos debería aparecer una lista filtrada como la siguiente

```
Choose a number or apply filter (format: [groupId:]artifactId, case
sensitive contains): : liferay
Choose archetype:
1: remote -> com.liferay.maven.archetypes:liferay-ext-archetype (Provides
an archetype to create Liferay extensions.)
```

```
2: remote -> com.liferay.maven.archetypes:liferay-hook-archetype
(Provides an archetype to create Liferay hooks.)
3: remote -> com.liferay.maven.archetypes:liferay-layouttpl-archetype
(Provides an archetype to create Liferay layout templates.)
4: remote -> com.liferay.maven.archetypes:liferay-portlet-archetype
(Provides an archetype to create Liferay portlets.)
5: remote -> com.liferay.maven.archetypes:liferay-portlet-icefaces-
archetype (Provides an archetype to create Liferay ICEfaces portlets.)
6: remote -> com.liferay.maven.archetypes:liferay-portlet-jsf-archetype
(Provides an archetype to create Liferay JSF portlets.)
7: remote -> com.liferay.maven.archetypes:liferay-portlet-liferay-faces-
alloy-archetype (Provides an archetype to create Liferay Faces Alloy
portlets.)
8: remote -> com.liferay.maven.archetypes:liferay-portlet-primefaces-
archetype (Provides an archetype to create Liferay PrimeFaces portlets.)
9: remote -> com.liferay.maven.archetypes:liferay-portlet-richfaces-
archetype (Provides an archetype to create Liferay RichFaces portlets.)
10: remote -> com.liferay.maven.archetypes:liferay-servicebuilder-
archetype (Provides an archetype to create Liferay Service Builder
portlets.)
11: remote -> com.liferay.maven.archetypes:liferay-theme-archetype
(Provides an archetype to create Liferay themes.)
12: remote -> com.liferay.maven.archetypes:liferay-web-archetype
(Provides an archetype to create Liferay webs.)
13: remote -> com.vaadin:vaadin-archetype-liferay-portlet (This archetype
creates a self-contained Vaadin Liferay portlet.
        It packages all Vaadin static resources (widgetset, theme etc.)
```

En negrita aparece el arquetipo en el que estamos interesados, el número 2, correspondiente a *'Liferay Hooks'*

Tecleamos el número 2 y pulsamos intro.

El siguiente paso es escoger el número de versión de *Liferay* para la que queremos crear el *Hook*. De la misma forma que con el arquetipo tecleamos el número de versión de *Liferay* y pulsamos intro.
A continuación debemos especificar el identificador del grupo del artefacto. Normalmente se usa la nomenclatura de dominio inverso, como podría ser por ejemplo:

```
es.carm.liferay.hook
```

Los dos últimos puntos es especificar el nombre y versión de nuestro artefacto. Ponemos los valores que deseemos.
Cuando hemos completado todos los pasos tenemos un plugin de tipo *Hook* con la estructura de proyecto Maven listo para empezar a trabajar sobre ella

Implementación de la acción

Para crear una acción *Liferay* basta con tener una clase *Java* en alguno de los paquetes del *hook* que herede de la clase

```
com.liferay.portal.kernel.events.Action
```

Al extender esta clase estamos obligados a implementar el método

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

Este es el método que *Liferay* invocará cuando tengamos nuestro *Hook* correctamente instalado

Definición de la acción

Por último nos queda indicarle a *Liferay* que ese *Hook* implementa una acción, de qué tipo y en qué clase se implementa la acción. Todo esto se hace mediante un fichero *.properties* que debe estar ubicado dentro del *CLASSPATH* del WAR que contendrá el Hook
Como estamos siguiendo la estructura de proyecto Maven podemos ubicar este fichero de propiedades en

```
/src/main/resources
```

Maven en tiempo de compilación se encargará de colocar el fichero de propiedades.
Una vez creado el fichero de propiedades dentro de esa ruta lo editamos. Dependiendo de qué evento queremos escuchar y cuando queremos ejecutarlo deberemos especificar alguna de estas variables:

- **servlet.service.events.pre**
- **servlet.service.events.post**
- **login.events.pre**
- **login.events.post**
- **logout.events.pre**
- **logout.events.post**

El valor que hay que asignarle a estas propiedades es la ruta completa de nuestra clase acción que queremos que se ejecute ante el evento.
Así que si queremos, por ejemplo que después de que un usuario se autentique se ejecute nuestra clase *es.carm.liferay.hook.actions.LoginAction* debemos establecer lo siguiente en el fichero de propiedades

```
login.events.post=es.carm.liferay.hook.actions.LoginAction
```

Por último debemos especificar a *Liferay* que debe leer este fichero de propiedades que va a ir dentro del *WAR*. Para eso debemos editar el fichero

```
src/main/webapp/WEB-INF/liferay-hook.xml
```

Suponiendo que nuestro fichero de propiedades se llama *mi-hook.properties* deberíamos dejar el fichero *liferay-hook.xml* así:

```
<?xml version="1.0"?>
<!DOCTYPE hook PUBLIC "-//Liferay//DTD Hook 6.2.0//EN" "http://
www.liferay.com/dtd/liferay-hook_6_2_0.dtd">

<hook>
      <portal-properties>mi-hook.properties</portal-properties>
</hook>
```