

Création d'une portlet avec support du framework Spring pour liferay

Introduction

Nous allons prendre l'exemple petportal proposé par spring, ajouter les XML manquants pour qu'il puisse répondre aux exigences de liferay, puis ajouter notre portlet exemple à l'application.

On suppose que vous avez déjà téléchargé spring 2.5.1 (version utilisé pour ce tutorial) avec ses dépendances.

Tout au long de ce document notre application aura comme nom XXXXX

Ajout des fichiers manquants.

Les fichiers manquants sont :

- liferay-portlet.xml pour lister les différents portlets.
- liferay-plugin-package.properties pour définir les propriétés de notre application
- liferay-display.xml pour la disposition de nos portlets dans le menu "add application" de liferay.

Les fichiers en question sont proposés avec ce document.

Création de la portlet

Définition dans portlet.xml

Chaque portlet doit être définie dans ce fichier XML en utilisant les balises suivantes :

- Portlet-name : Identificateur de la portlet qui doit être unique dans l'application.
- Portlet-class : La class dispatcher de notre application Spring.
- Vous pouvez la définir vous-même ou faire appel à une classe générique proposé par Spring (org.springframework.web.portlet.DispatcherPortlet).

- Init-param : paramètre pour l'initialisation de notre application. Nous allons juste ajouter contextConfigLocation et le faire pointer sur notre nouveau fichier.
- Supports : donner le type de la sortie (text/html pour notre cas) et les modes que la portlet va assurer (view pour notre cas)
- Portlet-info : pour donner des informations sur la portlet, ce que nous retiendrons c'est title pour définir le titre qui sera affiché pour les utilisateurs finaux.

```
<portlet>
    <portlet-name>XXXXX</portlet-name>
    <portlet-class>
        org.springframework.web.portlet.DispatcherPortlet
    </portlet-class>
    <init-param>
        <name>contextConfigLocation</name>
        <value>/WEB-INF/context/portlet-XXXXX.xml</value>
    </init-param>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>view</portlet-mode>
    </supports>
    <portlet-info>
        <title>Recherche client</title>
    </portlet-info>
</portlet>
```

Définition dans liferay-portlet.xml

Dans ce fichier on va juste faire une copie de notre ID du fichier précédant comme suit :

```
<portlet>
    <portlet-name>XXXXX</portlet-name>
</portlet>
```

Définition dans liferay-display.xml

Il ne reste plus qu'à ajouter notre portlet dans la catégorie qu'on voudrait la voir dans le bloc display comme suit :

```
<category name="YYYYY">
    <portlet id="XXXXX" />
</category>
```

Création du fichier context

L'emplacement de notre fichier a été défini dans portlet.xml, donc on va le créer sous WEB-INF, context, avec le nom portlet-XXXXX.xml.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

    <!-- Ici vont se trouver les définitions de nos beans -->

    <bean id="portletModeHandlerMapping"

        class="org.springframework.web.portlet.handler.PortletModeHandlerMapping">

        <property name="order" value="20" />

        <property name="portletModeMap">

            <map>

                <entry key="view" value-ref="xcontroleur" />

            </map>

        </property>

    </bean>

    <bean id="parameterMappingInterceptor"

        class="org.springframework.web.portlet.handler.ParameterMappingInterceptor" />

    <bean id="portletModeParameterHandlerMapping"

        class="org.springframework.web.portlet.handler.PortletModeParameterHandlerMapping">

        <property name="order" value="10" />

        <property name="interceptors">

            <list>

                <ref bean="parameterMappingInterceptor" />

            </list>

        </property>

    </bean>
```

```
</property>
<property name="portletModeParameterMap">
  <map>
    <entry key="view">
      <map>
        <entry key="xview"
          value-ref="xcontroleur" />
      </map>
    </entry>
  </map>
</property>
</bean>
<bean id="xcontroleur"
  class="org.tunisiana.controller.XControleur">
</bean>
</beans>
```

Ce que nous devons retenir ici c'est la définition de notre contrôleur et de son emplacement. Mais aussi, la partie View de l'application.

Création du contrôleur

Tout d'abord, il faut respecter l'emplacement attribué au début.

Un contrôleur Spring doit hériter de `AbstractController` qui se trouve sous `org.springframework.web.portlet.mvc.AbstractController`.

Un contrôleur doit définir les méthodes :

- `protected void handleActionRequestInternal(ActionRequest request, ActionResponse response)`
- `protected ModelAndView handleRenderRequestInternal(RenderRequest request, RenderResponse response)`

Remarques

L'appel de ses deux méthodes se fait successivement d'où si vous avez besoin de paramètre de formulaire dans la deuxième méthodes, il faudrait penser à les passer ou les faire passer toutes en utilisant :

```
response.setRenderParameters(request.getParameterMap());
```

Reste

Le reste est semblable à la création d'application Spring.