Généralités sur les points d'extension

Documentation nuxeo

Nuxeo propose de nombreux points d'extension. La liste et la description de ces points d'extension est disponible sur le site de nuxeo

Etendre un point d'extension

Nous avons vu dans le fichier de description XML d'exemple que l'utilisation d'un point d'extension existant se faisait via la balise /component/extension. Nous avons aussi vu que les balises filles de **/component/extension** étaient passées en paramètre au point d'extension étendu. Si la documentation nuxeo des points d'extension n'est pas suffisante il peut-être intéressant de retrouver cette information directement dans le code source de nuxeo.

Regardons sur l'exemple suivant :

```
<extension target="org.nuxeo.ecm.core.schema.TypeService"
   point="schema">
   <schema name="ori" src="schemas/ori.xsd"  prefix="ori" />
</extension>
```

Comment savoir que **org.nuxeo.ecm.core.schema.TypeService** attend **schema** comme balise fille ?

Démarche :

- Recherche d'un fichier de description XML dans le source de nuxeo contenant de **name="org.nuxeo.ecm.core.schema.TypeService"**
- Vérifier que ce service contient un **extension-point** de **name="schema"**. Si ce n'est pas le cas il faut rechercher un autre fichier de description XML qui définit ce point d'extension.
- Recherche, dans ce point d'extension, de la balise fille **object** dont on extrait l'attribut **class** qui est le bean de configuration utilisé par le point d'extension. Soit **org.nuxeo.ecm.core.schema.SchemaBindingDescriptor** dans notre exemple.
- Recherche du code source de la classe pour le lire :

```
@XObject("schema")
public class SchemaBindingDescriptor \{
    @XNode("@name")
    public String name;
    @XNode("@src")
    public String src;
    @XNode("@prefix")
    public String prefix = "";
    @XNode("@override")
    public boolean override = false;
```

Explications

@XObject("schema")	La balise fille à introduire dans le fichier de description XML de l'utilisation du point d'extension devra être **<schema>**
@XNode("@name") **public** String name;	La balise **<schema>** aura un attribut **name**. Il n'y a pas de valeur par défaut.
@XNode("@prefix") **public** String prefix = ""; 	La balise **<schema>** aura un attribut **prefix**. La valeur par défaut est une chaine de caractères vide.

Si on veut une explication plus précise sur l'utilité d'un attribut comme **src** par exemple. Il faut en regarder l'usage dans **org.nuxeo.ecm.core.schema.TypeService**.

On peut imaginer un point d'extension qui prend en paramètre, via son bean de configuration une classe java et que ce point d'extension la charge dynamiquement. C'est ce qui est utilisé par le point d'extension de type listener dans nuxeo

Créer un point d'extension

Nous avons vu dans le fichier de description XML d'exemple la définition d'un point d'extension de **name wsUrl**.

Voyons la marche à suivre :

- Définition d'un bean de configuration (**WsDescriptor** ici)
- Définir une interface (**OriOaiWorkflowService** ici)
- Définir ensuite une classe qui implémente cette interface (**OriOaiWorkflowServiceMock** ici). Cette classe :
    - Etend org.nuxeo.runtime.model.DefaultComponent
    - Surcharge la méthode **registerContribution** ce qui lui permet de récupérer un objet de type ComponentInstance qui une fois casté pointe vers notre bean de configuration. NB : Si le point d'extension est utilisé n fois, cette méthode sera aussi appelée n fois.

```
@Override
public void registerContribution(Object contribution,
        String extensionPoint, ComponentInstance contributor) \{
    config.add((WsDescriptor) contribution);
}
```

- - Surcharge aussi **unregisterContribution** pour l'opération inverse
    - Elle aurait aussi pu surcharger (cf. **OriOaiNuxeo2XmlServiceImpl** pour exemple) :
        - **registerExtension** qui est appelée une fois par point d'extension même si ce dernier est utilisé n fois. Ceci peut-être utile pour agréger les informations collectées lors des n appels à **registerContribution** par exemple.
        - Activate qui est appelé quand le point d'extension est activé ce qui permet, par exemple, de récupérer le contexte d'exécution du point d'extension.

Notre point d'extension étant créé et configurable via son bean de configuration il va pouvoir être étendu à son tour. Exemple de **wsdl-orioaiworkflow-contrib.xml** qui étend ce point d'extension en précisant une configuration :

```
<component name="org.orioai.nuxeo.workflow.wsdlOriOaiWorkflow">
  <extension target="org.orioai.nuxeo.workflow.OriOaiWorkflowService"
    point="wsUrl">
    <configuration>
      <wsUrl>http://localhost:6580/ori-oai-workflow-spring/xfire/OriWorkflowService</wsUrl>
    </configuration>
  </extension>
</component>
```

Il va aussi être possible de faire appel aux services de notre point d'extension depuis les contrôleurs liés aux vues de notre « application » contenue dans nuxeo. C'est ce que nous allons voir dans les chapitres suivants.