Principaux fichiers d'un plugin nuxeo

MANIFEST.MF

> ⚠ Fichier src/main/resources/META-INF/MANIFEST.MF dans le projet

Le fichier **MANIFEST.MF** est le fichier de configuration principal d'un plugin OSGI.

Exemple :

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 1
Bundle-Name: Nuxeo Sample project
Bundle-SymbolicName: org.orioai.nuxeo.workflow;singleton:=true
Bundle-Version: 1.0.0
Bundle-Vendor: ORI-ORI.org
Provide-Package: org.orioai.nuxeo.workflow
Require-Bundle: org.nuxeo.runtime,
 org.nuxeo.ecm.core.api,
 org.nuxeo.ecm.core,
 org.nuxeo.ecm.webapp.core
Nuxeo-Component: OSGI-INF/core-types-contrib.xml,
 OSGI-INF/actions-contrib.xml,
 OSGI-INF/orioaiworkflow-service-contrib.xml,
 OSGI-INF/wsdl-orioaiworkflow-contrib.xml,
 OSGI-INF/orioainuxeo2xml-service-contrib.xml,
 OSGI-INF/xsl-orioainuxeo2xml-contrib.xml
Bundle-ClassPath: qname.jar
Import-Package: javax.xml.namespace
```

Explications

Information générale sur le plugin

Bundle-SymbolicName	ID
Bundle-Version	Version
Bundle-Name	Nom
Bundle-Vendor	Fournisseur

Gestion de dépendances

Provide-Package	Chaîne de caractère qui devra être utilisé par des plugin qui seront dépendant de ce plugin
Require-Bundle	Liste des plugins dont est dépendant le plugin courant. Cette liste influera sur l'ordre de chargement des plugins au démarrage de nuxeo

Autres paramètres

Nuxeo-Component	Autres fichiers, appelés XML descriptor, de configuration du plugin. Ils servent à spécifier les points d'extension utilisés par le plugin et/ou à spécifier les points d'extension exposés par le plugin
Bundle-ClassPath	Liste de répertoires ou de jar à charger pour la bonne exécution du plugin
Import-Package	???

XML descriptor

Fichiers src/main/resources/OSGI-INF/*-contrib.xml dans le projet

Ce sont les fichiers spécifiés dans **Nuxeo-Component** du fichier **MANIFEST.MF**

Ils servent à spécifier les points d'extension utilisés par le plugin et/ou à spécifier les points d'extension exposés par le plugin.

Par convention ils sont de la forme :

- ***-contrib.xml** quand ils servent à spécifier l'utilisation de points d'extension existants
- ***-service-contrib.xml** quand ils servent à spécifier de nouveaux points d'extension

Le développement des points d'extension est traité dans ??? mais voici 2 exemples de fichiers XML descriptor.

Exemple de core-types-contrib.xml :

```
<?xml version="1.0"?>
<component name="org.orioai.nuxeo.workflow.core-types">
  <extension target="org.nuxeo.ecm.core.schema.TypeService"
    point="schema">
    <schema name="ori" src="schemas/ori.xsd"  prefix="ori" />
  </extension>
  <extension target="org.nuxeo.ecm.core.schema.TypeService"
    point="doctype">

    <doctype name="File" extends="Document">
      <schema name="common" />
      <schema name="file" />
      <schema name="dublincore" />
      <schema name="uid" />
      <schema name="files" />
      <schema name="ori" />
      <facet name="Downloadable" />
      <facet name="Versionable" />
      <facet name="Publishable" />
      <facet name="Indexable" />
    <facet name="Commentable" />
    </doctype>
  </extension>
</component>
```

Explications

/component@name	Nom du composent. Ce nom prend la forme d'une classe java mais il ne s'agit pas obligatoirement d'une classe. En effet, le plus souvent, on étend un point d'extension pour lui passer des paramètres de configuration.
/component /extension@target	Nom du composant que l'on va étendre. Correspond au /component@name du point d'extension utilisé.
/component /extension@point	Un composant peut contenir plusieurs points d'extension utilisables. On spécifie avec ce paramètre celui que l'on veut étendre.
/component /extension/*	Liste des informations que l'on va passer un point d'extension que l'on étend. Cette liste est spécifique à chaque point d'extension.

Exemple de orioaiworkflow-service-contrib.xml :

```
<?xml version="1.0"?>
<component name="org.orioai.nuxeo.workflow.OriOaiWorkflowService">
      <implementation
            class="org.orioai.nuxeo.workflow.OriOaiWorkflowServiceMock" />
      <service>
            <provide
                  interface="org.orioai.nuxeo.workflow.OriOaiWorkflowService" />
      </service>
      <documentation>explication sur le service</documentation>
      <extension-point name="wsUrl">
            <documentation>explication sur le point d'extension</documentation>
            <object class="org.orioai.nuxeo.workflow.WsDescriptor" />
      </extension-point>
</component>
```

Explications

/component@name	Nom du composent. Ce nom prend la forme d'une classe java. Une bonne pratique consiste à utiliser ici la même valeur que pour /component/service /provider@interface
/component /implementation@class	Classe java concrète implémentant le service
/component/service /provider@interface	Nom de l'interface java qui est implémentée par /component/implementation@class
/component /documentation	Texte d'information sur le service. Il est utile pour les personnes qui seront amenés à travailler avec votre point d'extension. Il est utilisé pour la génération automatique de la documentation de référence sur les points d'extension.
/component/extension-point@name	Nom du point d'extension offert par le composant. Il peut y avoir plusieurs balises /component/extension-point
/component/extension-point/documentation	Texte d'information sur le point d'extension
/component/extension-point/object	Nom du bean qui contiendra les configurations du point d'extension lorsqu'il sera étendu.

Déploiement fragment

> ⚠ Fichiers src/main/resources/OSGI-INF/deployment-fragment.xml dans le projet

Une application web java classique a des fichiers de configuration qui sont lus au lancement du serveur d'applications. De même, les fichiers de ressources ou les librairies sont lus au lancement. Ceci n'est pas compatible avec la notion de plugin.

En effet, un plugin se matérialise par un simple fichier jar qui sera chargé en fonction des règles de dépendances définies dans le fichier MANIFEST.MF. Il peut néanmoins nécessiter des ressources spécifiques ou une configuration particulière de l'application java qui la contient.

Le fichier deployment-fragment.xml permet justement de spécifier comment le plugin va adapter l'environnement d'exécution java dans lequel il sera appelé.

Exemple :

```
<?xml version="1.0"?>
<fragment>
  <extension target="application#MODULE">
    <module>
      <java>$\{bundle.fileName}</java>
    </module>
  </extension>
  <extension target="web#CONTEXT-PARAM">
    <context-param>
      <param-name>org.jboss.seam.core.init.debug</param-name>
      <param-value>true</param-value>
    </context-param>
  </extension>
  <install>
    <!- Unzip the war template ->
    <unzip from="$\{bundle.fileName}" to="/">
      <include>nuxeo.war/**</include>
    </unzip>
    <!- Append .properties files ->
    <delete path="orioainuxeodir.tmp" />
    <mkdir path="orioainuxeodir.tmp" />
    <unzip from="$\{bundle.fileName}" to="orioainuxeodir.tmp">
      <include>*/.properties</include>
    </unzip>
    <append from="orioainuxeodir.tmp/OSGI-INF/l10n/messages.properties"
      to="nuxeo.war/WEB-INF/classes/messages.properties"
      addNewLine="true" />
    <append from="orioainuxeodir.tmp/OSGI-INF/l10n/messages_fr.properties"
      to="nuxeo.war/WEB-INF/classes/messages_fr.properties"
      addNewLine="true" />
    <delete path="orioainuxeodir.tmp" />
  </install>
</fragment>
```

Explications :

/fragment /extension@target	Nuxeo utilise des fichiers templates (dans le répertoire nuxeo.ear/OSGI-INF/templates). la propriété /fragment/extension@target permet de cibler ce fichier (partie avant le # sans l'extension .xml) ainsi que le positionnement dans ce fichier (partie après le # qui permet de pointer sur une balise de la forme %{MODULE}%)
/fragment /extension/*	Portion de XML qui sera intégrée dans le fichier template
/fragment /install	Contient du code de type ant permettant de faire certaines actions sur le système de fichiers de l'application. Ici : - on copie le contenue du répertoire src/main/resources/nuxeo.war de notre projet vers le répertoire d'exécution de l'application java - On vient ajouter des chaines de caractère dans le fichier de message de l'application
Note à propos de ${bundle.fileName}	${bundle.fileName} correspond au fichier jar du plugin courant

Nuxeo.war

> ⚠ répertoire src/main/resources/nuxeo.war dans le projet

Ce répertoire contient l'ensemble des fichiers (pages JSP en .xtml, image, etc.) qui seront déployés dans le répertoire d'exécution de l'application Web de nuxeo grâce au deployment-fragment.xml.