



UNIVERSITÉ DE NANTES

Master 1 ALMA

Projet de Conception de Logiciels Extensibles

Plateforme de plugins - Architecture et fonctionnement

Étudiants :

Coraline MARIE, Vincent RAVENEAU et Quentin MORICEAU

Encadrant :

Gilles ARDOUREL

30 mars 2014

Table des matières

1	Architecture de la plateforme	2
1.1	Package <code>interfaces</code>	2
1.2	Package <code>main</code>	3
1.3	Package <code>pluginManager</code>	3
	Plan :	
	Intro	
	but du projet	
	cahier des charges	
	Dépendances externes	
	Architecture	
	Plugins indispensables	
	Création de données	
	Affichage de données	
	Modification de données	
	Conclu	
	ce qu'il manque (éditer les données)	
	(création par l'interface graphique)	

Architecture de la plateforme

La plateforme à plugins, fournie dans le projet Eclipse `GestionnairePlugin`, s'articule autour de 3 packages, dont le contenu sera détaillé dans les parties suivantes.

1.1 Package interfaces

Ce package contient uniquement des interfaces, utilisées par d'autres projets souhaitant utiliser les fonctionnalités de la plateforme de plugins. Ces interfaces sont les suivantes :

- **`interfaces.IPlugin`** : Cette interface permet à la plateforme d'identifier une classe Java comme un plugin valide. Elle est destinée à être implémentée par les plugins basiques, ne nécessitant pas le chargement d'autres plugins pour fonctionner. Une fois implémentée, elle impose des méthodes permettant à la plateforme d'exécuter le plugin, de récupérer son type, et de lui transmettre un objet contenant les informations trouvées dans le fichier de configuration donné lors de la demande de chargement du plugin.
- **`interfaces.IComplexPlugin`** : Cette interface, qui étend l'interface **`interfaces.IPlugin`** précédemment décrite, est destinée à être implémentée par les plugins complexes ayant eux-même besoin de charger des plugins pour fonctionner. En plus des méthodes d'**`interfaces.IPlugin`**, elle impose une méthode permettant à la classe gérant la création des plugins de transmettre son instance aux plugins complexes créés. Ainsi, ces plugins seront en mesure de faire appel au gestionnaire de plugins pour charger tout ce dont ils ont besoin pour s'exécuter.
- **`interfaces.IPluginManager`** : Cette interface permet d'identifier une classe comme étant un gestionnaire de plugins. Utilisée par les plugins complexes pour demander le chargement des plugins dont ils ont besoin, elle impose pour ce faire 2 méthodes. L'une permet de demander un plugin précis, en donnant le nom de la classe Java associée et le chemin vers son fichier d'initialisation. L'autre permet de demander le chargement d'un plugin aléatoire d'un type donné.

1.2 Package main

Ce package est composé d'une unique classe Java, `main.PluginsPlatform`. Cette classe contient la méthode `main` à exécuter pour démarrer la plateforme, qui va créer une instance de la classe `pluginManager.PluginManager`, qui constitue le coeur de la plateforme.

1.3 Package pluginManager

Ce package contient la classe `pluginManager.PluginManager`, qui est la classe gérant le chargement et l'exécution de tous les plugins de la plateforme. Implémentant l'interface `interfaces.IPluginManager` précédemment décrite, son comportement repose sur le contenu du fichier d'initialisation qui lui est donné (`GestionnairePlugins/resources/init`).

Ce fichier est une suite de paires *clef/valeur*, destinées à être récupérées par une instance de `java.util.Properties`. Afin d'avoir un fonctionnement correct de la plateforme, les 3 clefs suivantes doivent avoir une valeur :

- `loadAtStart` : Contient le nom complet des plugins devant être chargés au lancement de la plateforme. Dans notre cas, il s'agit de l'éditeur d'armure.
- `homePath` : Contient le chemin relatif depuis le répertoire `$home` de la machine exécutant la plateforme vers un répertoire regroupant tous les plugins susceptibles d'être demandés lors de l'exécution de la plateforme. Les sous-dossiers étant pris en compte, tout comme les liens relatifs, il n'est pas nécessaire de!!!!!!!!!!!!!!!!!!!! A REFORMULER, erreur dans l'explication (`homePath/initPath`)!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!