UNIVERSITÉ DE NANTES

Master 1 ALMA

Projet de Conception de Logiciels Extensibles

Documentation Développeur

Étudiants :
Coraline MARIE, Vincent RAVENEAU et Quentin MORICEAU

Encadrant :
Gilles ARDOUREL

30 mars 2014

Table des matières

Préface

L'Éditeur d'Armure est un logiciel obéissant à un scénario composé par les étudiants du Master 1 Alma : un utilisateur quelconque souhaîte créer ou éditer des données, décrivant une armure de protection/combat (telle celle d'Iron Man).

Ce projet, à but pédagogique, a été conçu afin de permettre aux étudiants de parfaire leurs connaissances en *Conception de logiciels extensibles* en créant leur propre architecture à plugins.

1.1 Récupération du code source

Le code source de l'application est disponible sur un dépôt Git hébergé par le site GitHub. Pour le moment, le code source peut être visualisé et télécharger par tout le monde. Il est possible de le récupérer, sous forme d'archive, ou directement par l'intermédiaire de Git.

```
git clone https://github.com/Lerian/ProjetCLE.git
```

1.2 Importation et architecture

L'Éditeur d'Armure est un logiciel conçus en `Java 7` avec l'IDE Eclipse. Pour accéder au code source sans soucis, il est conseillé d'importer au préalable les projets sous Eclipse avant d'y faire des modifications.

Le code source est divisé en plusieurs projets :

- Application projet : plugin principal qui contient le plugin ArmorEditor ;
- GestionnairePlugins : gestionnaire de plugin indispensable au bon fonctionnement du logiciel ;
- CreationArmure : plugin de création automatique d'armure ;
- CreationArmureFichier : plugin de création d'armure, à partir d'un fichier texte ;
- ModificationArmure : plugin qui contient un ensemble de méthodes, permettant de modifier les données d'une armure ;
- AffichageConsole : plugin d'affichage textuel dans la console ;
- AffichageGraphique : plugin d'affichage graphique.

Description des projets fournis

L'Éditeur d'Armure est un plugin qui se base sur un gestionnaire de plugins et sur plusieurs autres plugins pour fonctionner.

2.1 GestionnairePlugins et ApplicationProjet

Le gestionnaire de plugin sert à gérer tous les plugins (y compris l'ArmorEditor). Son comportement et son architecture sont détaillés dans le document concernant l'architecture. Le gestionnaire de plugin permet de lancer l'application, et aussi de charger des plugins.

L'ApplicationProjet, quant à lui, est le projet principal du logiciel. Il contient le plugin ArmorEditor, les classes qui permettent de décrire une armure, et les interfaces nécéssaires à l'ArmorEditor pour communiquer avec les plugins dont il a besoin (afficheur, modificateur ...).

2.2 Plugins d'affichage

Les plugins d'affichage regroupent tous les plugins qui implémentent l'interface `IAfficheur`. Deux d'entre eux sont fournis actuellement.

Le plugin `AffichageConsole`, qui utilise simplement la methode `toString` sur l'instance de l'armure, et qui affiche les informations recueillies dans la console utilisée pour exécuter le logiciel. Le plugin `AffichageGraphique`, plus complexe, qui annalyse les données d'une armure pour les afficher proprement dans une fenêtre.

2.3 Plugins de création

Les plugins de création regroupent tous les plugins qui implémentent l'interface `ICreateur`. Deux d'entre eux sont fournis actuellement.

Le plugin `CreationArmure` permet d'instancier une armure de base (décrite directement dans le code). Le plugin `CreationArmureFichier` prend un fichier texte (`.txt`) en paramêtre, et récupére les propriétés décrites dans ce fichier pour instancier une armure. Le format du fichier texte est décrit plus précisément dans le tutoriel présent dans la documentation utilisateur.

2.4 Plugin de modification

Les plugins de modification regroupent tous les plugins qui implémentent l'interface `IModificateur`. Celui que nous avons créé contient un ensembles de méthodes capables de modifiées les propriétés d'une armure instanciée.

Tutoriel : comment créer un plugin

Pour créer un nouveau plugin il est conseillé d'utilisé l'IDE Eclipse et d'y importer les projets disponible sur GitHub par la commande :

```
git clone https://github.com/Lerian/ProjetCLE.git
```

Ou en passant par le plugin git sous eclipse.

Une fois les projets correctements importé, vous pouvez commencé la création de votre plugin.

Pour commencé, créez un nouveau projet java.

Créez ensuite une classe qui sera votre classe principal. Cette classe devra implémenter l'interface IPlugin du projet GestionnairePlugins.

Si vous souhaitez creez un plugin pour le logiciel d'édition d'armure, vous devez choisir le type de plugin à creez :
- Créateur : implémenter l'interface ICreateur du projet ArmorEditor.
- Modificateur : implémenter l'interface IModificateur du projet ArmorEditor.
- Afficheur : implémenter l'interface IAfficheur du projet ArmorEditor.

Si des erreurs apparaisse ajouter les projets GestionnairePlugins et ArmorEditor à votre build path.

Vous pouvez maintenant créer votre plugin en implémentant les méthodes des interfaces.

La méthode run est executé lors de l'ajout de votre plugin au lancement de la plateforme.
La méthode type doit retourné le type du plugin qui est à définir, pour l'éditeur d'armure, il existe déjà 4 types défini dans l'enum PluginTypes : CREATEUR, AFFICHEUR, MODIFICATEUR et MAIN.
La méthode receiveProperties fournie un objet Properties qui contient : "pathToHome" qui retourne sous forme de string le chemin d'accès vers le dossier dans lequel est contenut le projet GestionnairePlugins. Mais aussi "pathToInit" qui retourne sous forme de string le chemin d'accès vers le fichier .init de votre

projet qui est à créer. Dans ce fichier.init vous pouvez inclure toute les informations que vous désirez récupérer dans votre code via l'objet Properties.

En ce qui concerne le programme d'édition d'armure, si vous souhaitez créer un afficheur, vous devez mettre le code principal de votre affichage dans la méthode affiche().

Pour le modificateur vous devez implementer les différentes méthodes qui permettents de changé les valeurs des armes et armures.

Et enfin pour le créateur, vous devez construire votre armure dans la méthode cree().

Pour que votre plugin soit executé au démarage de la plateforme, vous devez indiquez la classe principal de votre plugin dans le champs loadAtStart.

Par contre si vous souhaitez construire un nouveau composant pour l'éditeur d'armure, vous devez remplir le fichier armorEditor.init en suivant la syntaxe décrite pour que votre plugin puisse être utilisé par le programme d'édition d'armure.

Voila vous avez créer votre plugin, félicitation !