DUBOIS Sébastien
HEUZE Sébastien
ROLLAND Baptiste
NASSIRI Khalid

Guide Développeur

Projet de plate-forme à plugins

Master 2 MIAGE, 2014/2015

Préambule

Le présent document présente l'ensemble des étapes nécessaires à l'utilisation et au développement de plugins en utilisant la plateforme de plugins développé dans le cadre d'un projet de travaux pratiques dans le module "Architecture logicielle" en Master 2 MIAGE.

Ce document se présente comme un tutoriel détaillé permettant la création de plugins pour le jeu Age Of Mottu utilisant cette plateforme

I – Création d'un plugin

A) Introduction

Pour créer un projet plugin il est important de comprendre les données qui le caractérisent :

- L'interface qu'il va implémenter : Elle va permettre au plugin principal d'utiliser ce plugin en suivant cette interface, chaque plugin doit se conformer à cette interface afin d'être employable dynamiquement par n'importe quel plugin.
  Les interfaces sont au nombre de 4 :
  - **IPlugin** : Interface globale des plugins, qui va être implémentée par TOUS les plugins, elle va permettre le chargement de ces derniers
  - **IActionPlugin** : Interface pour les plugins d'actions, une action sera déclenchable via la méthode doAction
  - **IDisplayPlugin** : Interface pour les plugins d'affichage
  - **ILauncherPlugin** : Interface pour le plugin principal

- Le type de plugin, voici les types de plugins actuellement disponibles, cette liste pourra être complétée au fur et à mesure :
  - **AFFICHAGE** : Plugin permettant l'affichage d'un composant au centre de l'application (Boutons situés à droite de l'application)
  - **ACTION** : Plugin permettant d'effectuer une action (Boutons en bas de l'application)
  - **AFTER**_ACTION : Plugin exécuté après chaque action
  - **LANCEUR** : Plugin principal

B) Etapes de création

Etape 1 : Création du Projet Eclipse

Créer un nouveau Projet JAVA sur Eclipse à coter de ceux déjà existants

Nommez votre projet par exemple PluginArc, sélectionnez une JRE > 1.7 puis faire Suivant => Terminer

Etape 2 : Création de la classe

Créer un nouveau package nommé plugin (Clic droit => Nouveau => Package) que vous allez nommer plugin

Une fois ce package crée, créez dans ce dernier une classe nommée Arc.java

Etape 3 : Configuration du Build Path

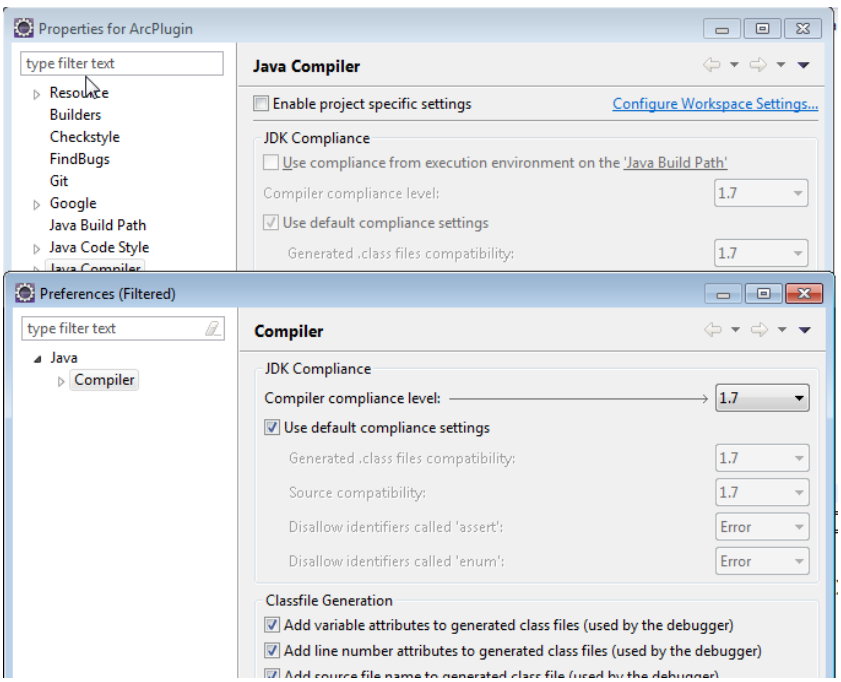
Notre projet aura temporairement besoin qu'on lui indique où trouver certaines classes JAVA (Interfaces, classes métier) il va donc falloir dire à Eclipse ou elles se trouvent

Faites un clic droit sur le projet => Build Path => Configurer le Build Path

Dans l'onglet Projet cliquez sur ajouter et ajoutez le projet PluginLanceur (Plugin principal) et Projet_Jeu_Plugins.

Etape 4 : Vérification de la version de compilation JAVA

Pour éviter des projets du à la version de JAVA faites un clic droit sur le projet => Propriétés => Java Compiler puis vérifiez que le Enable Project Specific settings est bien désactivé, cliquez sur Configure Workspace Settings pour vérifier que le Workpace est bien configuré pour fonctionner en JAVA 1.7



Etape 5 : Création du code

Ouvrez le fichier PluginDemoArc.txt à la racine du workspace pour un exemple de code, et copier/coller le dans le Arc.java

Le principe de l'Arc est qu'il va pouvoir attaquer les ennemis deux fois plus loin que l'épée

Etape 6 : Génération du JAR

Pour générer automatiquement le jar nous allons créer un Build Ant (vu qu'il est intégré de base dans Eclipse)

Clic droit sur le projet => New => File Nommez le build.xml et ajoutez le code suivant :

```
<?xml version="1.0" ?>
<!-- Configuration of the Ant build system to generate a Jar file -->
<project name="TestMain" default="CreateJar">
  <target name="CreateJar" description="Create Jar file">
        <jar jarfile="../finalApp/mods/PluginArc.jar" basedir="bin"
includes="*/*.class"/>
             <jar jarfile="../Projet_Jeu_Plugins/mods/PluginArc.jar" basedir="bin"
includes="*/*.class"/>
  </target>
</project>
```

Cela va permettre de générer le jar dans le dossier d'application finale et dans le dossier Projet_Jeu_Plugins pour les exécutions eclipse

Et comme on est fainéants on va automatiser ça

Clic droit sur le projet => Propriétés => Builders => New => Ant Builder

Vous aller tomber sur un assistant de configuration, deux choses sont à faire :

1- Dans l'onglet Main, dans la première case BuildFile sélectionnez Browse Workspace et sélectionnez le build.xml (assurez-vous de bien prendre celui du bon projet)
2- Dans l'onglet Targets dans la 3ème case (Auto Build) sélectionnez Set Targets puis sélectionnez CreateJar (Sélectionné par défaut ne faite rien) => OK

Puis OK et OK encore, maintenant à chaque Sauvegarde/Clean un jar sera généré au bon endroit

Etape 7 : Ajout du plugin dans les fichiers configs

Ajoutez la ligne suivant pour activer le plugin dans les 2 fichiers config.txt (Dossier finalApp et Projet_Jeu_Plugins)

Arc = IActionPlugin;ACTION;PluginArc;true

Plugin terminé !

II – Quelques aides à la programmation

A) Accès à la liste de plugins

L'accès à la liste de plugin se fait par l'appel de :

```
Platform.getInstance().getPluginsInfo(IDisplayPlugin.class, TypePlugin.AFFICHAGE);
```

Qui renvoi une liste de PluginInfo, le premier paramètre prend l'interface du plugin et le deuxième prend le type de plugin.

B) Accès au plugin principal

L'accès au plugin principal se fait par l'appel de :

```
Platform.getInstance().getLauncherPlugin()
```

Qui renvoi un objet de type ILauncherPlugin (interface du plugin principal)
Il est ensuite possible d'appeler getObjectInstance(Hero.class) pour par exemple accéder à l'objet Hero du Launcher.

Ceci est un Getter générique qui permet d'accéder aux objets des couches inférieures.