Manuel pour les développeurs de plugins

Table des matières

1. Contexte

Le plugin à développer s'inscrit dans le cadre de Pivot. Ce manuel indique donc comment procéder au développement d'un plugin pour Pivot, et quelle est l'architecture à respecter.

2. Archive fournie aux développeurs de plugins

Des données nécessaires au développement d'un plugin pour Pivot sont fournies sous la forme d'une archive .jar.

Cette archive contient l'interface *Plugin* et la classe abstraite *Liaison* dans un package *plugin.core*, mais également un autre package nommé *plugin.definition* contenant des définitions de plugins spécifiques. Une classe *StarlistKeyword*, dans un package *pivot_client* et contenant des constantes représentant les différentes données (et leur indice) d'un starlist acceptées par l'application, est également présente pour les développeurs de plugins souhaitant connaître le format des starlists (et de ses données) acceptés par l'application.

Cette archive est disponible sur http://pivot-ws.oca.eu/

2.1. Explication des classes et interfaces fournies

L'interface *Plugin* permet de définir les méthodes communes à tous les plugins. Le plugin à développer ne doit pas implémenter directement cette interface, mais doit implémenter une interface (spécifique) du package *plugin.definition*. Sans cela, le plugin ne sera pas accepté par l'application.

La classe abstraite *Liaison* définit des méthodes de l'application Pivot que peut avoir besoin le plugin. Cette classe contient donc des méthodes que le plugin connait et peut appeler. Cependant, cette *Liaison* ne doit pas être utilisée directement par le plugin : il doit être utilisé une Liaison spécifique dans le package *plugin.definition* correspondant au plugin spécifique choisi ci-dessus.

Le plugin spécifique et la liaison spécifique se trouvent dans le même package.

2.2. Utilisation de l'archive fournie dans Pivot

Le constructeur par défaut (sans paramètre) du plugin développé est appelé par l'application. La méthode *initialize()* est ensuite appelé afin d'initialiser la liaison. Le plugin démarre ensuite lors de l'appel de la méthode *start()*. Le plugin reçoit alors les données dont il a besoin pour fonctionner. C'est également à ce moment-là que le panneau du plugin est récupéré et inclus dans un onglet ou une nouvelle fenêtre de l'application. À la fermeture de l'onglet ou de la fenêtre est appelée la méthode *stop()*. Et à la fermeture de l'application, la méthode *destroy()* est appelée.

Pour plus d'informations, consulter la Javadoc de l'archive fournie : http://pivot-ws.oca.eu/

3. Exploitation de l'archive fournie et création du plugin

Pour créer un plugin pour Pivot, un projet Java doit être créé sous Netbeans. La première des choses à faire est d'importer l'archive .jar fourni aux développeurs de plugins. Il faut vérifier que le numéro de version présent dans le nom de l'archive correspond bien au numéro de version de Pivot.

Pour ajouter un .jar à un projet java sous Netbeans, il faut faire un clic droit sur « Librairies », suivi d'un clic gauche sur « Add JAR/Folder… ». Il faut alors sélectionner l'archive est cliquer sur le bouton « Ouvrir ». L'archive est alors ajoutée au projet Netbeans.

Une fois le développement du plugin terminé sous Netbeans, il faut procéder à la création du .jar représentant le plugin. L'archive du plugin ne doit en aucun cas inclure celle fournie aux développeurs de plugins, sinon cela provoquera des erreurs lors de son chargement dans Pivot.

4. Prises en compte du plugin par Pivot

Le plugin, afin d'être pris en compte par l'application, doit se trouver dans un dossier « plugins » situé dans le même dossier que le .jar de l'application (plugin sous la forme d'un .jar). Si le plugin développé ne respecte pas les contraintes imposées ci-dessous, il ne sera pas pris en compte par l'application.

5. Contraintes à respecter

Le plugin doit être sous la forme d'une archive .jar.

Les classes fournies aux développeurs de plugins ne doivent pas être inclus dans le .jar final du plugin (car elles sont déjà présentes dans l'application).

Une seule et unique classe du plugin doit implémenter une interface *Plugin* spécifique présente le package *plugin.definition*.

Un plugin ne peut avoir qu'une spécificité, c'est-à-dire implémenter qu'une seule interface *Plugin* spécifique.

Le plugin, pour être pris en compte par l'application, doit se trouver dans un dossier « plugins » situé dans le même dossier que le .jar de l'application.