UFR Sciences et Techniques

« JARVIS, *a Java Agenda with ReflexiVe Interface System.* »

Conception de logiciels extensible – X8IA030

Réalisation :

Antoine Carat
Margaux Cherrueau
Sitraka Fidimihajamanana
Laurent Girard
Mathieu Groudin
Pierre Lemétayer

2016 - 2017

Sommaire

Résumé

JARVIS *est un gestionnaire d'agenda écrit en Java. Il permet la gestion d'une liste d'événements (création, modification, suppression, import, export, ...). L'application* JARVIS *est lancée par le biais d'une plateforme et utilise des extensions gérées par celle-ci.*

Chapitre 1 : Lancement et utilisation de l'application

Pour lancer notre application, il faut utiliser la commande *java -jar JARVIS.jar*.

Une fois l'application lancée, deux fenêtres s'affichent. La première *Agenda* JARVIS se décompose en 2 parties.

La première partie à gauche permettant la créations d'évènement. Nous avons 2 boutons, le premier *New Event* vous ouvre une fenêtre permettant de créer un évènement avec son nom, sa date de début et de fin, son type (*Anniversaire*, *Rendez-vous*, ...), une description ainsi qu'un lieu. Et le deuxième *Import* vous permet d'importer des événements à partir du fichier *.yaml* de votre choix.

La seconde partie à droite est composée d'un affichage des événements et de plusieurs boutons permettant des affichages différents. Nous avons la possibilité d'afficher de 4 façons différentes. La liste de tous les événements avec le bouton *All*, la liste des événements du mois courant avec le bouton *Month*, nous avons ensuite le bouton *Export* permettant d'exporter tous les événements de notre application dans un fichier *.yaml*. Et pour finir nous avons le bouton *Edit* permettant d'éditer ou bien de supprimer un événement existant.

Dans cet affichage, qui est celui par défaut, 2 boutons sont associés à chaque événement. Le premier *Edit* ouvre une fenêtre permettant de modifier un des champs de l'événement. Le second bouton *Delete* nous ouvre une fenêtre demandant la confirmation de suppression de cet événement.

La deuxième fenêtre *Monitoring* présente l'ensemble des extensions disponibles, leur état ainsi que le nombre d'instances. Nous voyons au démarrage de l'application que seulement 3 extensions sont en marche à savoir les extensions *Base*, *ModifierPrinter* et *Monitoring*. La *Base* correspond donc à la fenêtre *Agenda* JARVIS ainsi que les boutons de création et d'affichage. La *ModifierPrinter* correspond à la liste des évènements et leurs boutons. Pour finir, l'extension *Monitoring* correspond à la fenêtre du même nom.

Chapitre 2 : Description de l'application

2.1 Fonctionnement de la plateforme

Au lancement de la plateforme, le fichier *config.yaml* est lu afin de charger les différents descripteurs des extensions. Dans l'exemple ci-dessous du fichier *config.yaml* de notre application, 10 extensions sont à prendre en compte.

```
1  name: JARVIS
2  description: JARVIS is a framework that manages plugins.
3  extensions:
4    - plugins.simpleBase.Base
5    - plugins.exportPrinter.ExportPrinter
6    - plugins.importPrinter.ImportPrinter
7    - plugins.simpleCreator.SimpleCreator
8    - plugins.simplePrinter.SimplePrinter
9    - plugins.monthPrinter.MonthPrinter
10   - plugins.simpleModifier.SimpleModifier
11   - plugins.deleteEventModifier.DeleteEventModifier
12   - plugins.modifierPrinter.ModifierPrinter
13   - plugins.monitoring.Monitoring
```

Chacun de ces descripteurs contiennent les propriétés, l'état (*available*, *running* ou *failed*) et les instances de l'extension. Ces informations sont récupérées à partir du fichier de configuration de chaque extension placé dans le dossier *pluginConfig/*.

Après avoir chargé les différents descripteurs d'extensions, la plateforme va lancer les extensions *autorun*, c'est à dire les extensions qui sont lancées au lancement de la plateforme.

Les extensions *autorun* sont *Base* et *Monitoring*. L'extension *Monitoring* offre une vue d'ensemble des extensions disponibles sur la plateforme, leur état et le nombre d'instances en cours de fonctionnement. L'extension *Base*, quant à elle, est l'extension correspondante à l'application d'agenda, c'est elle qui va demander à la plateforme de lancer toutes les autres extensions.

Pour construire la fenêtre, l'extension *Base* demande à la plateforme la liste des extensions implémentant l'interface *ICreator* et affiche un bouton pour chacune des extensions. En cliquant sur le bouton, l'extension *Base* va demander à la plateforme de lancer l'extension correspondante.

Pour les différents afficheurs, l'extension *Base* va faire deux demandes successives : tout d'abord elle va demander les extensions implémentant l'interface *IPrinter* et ayant la propriété *default* positionnée à *vrai* et ensuite demander la liste d'extensions implémentant l'interface *IPrinter*. On aura donc ici un bouton

par afficheur et l'afficheur par défaut sera lancé.

Dans le plugin *modifierPrinter*, nous offrons la possibilité d'éditer un évènement au travers des extensions implémentant l'interface *IModifier*. Cet afficheur va demander à la plateforme les différents modifieurs et afficher un bouton pour chacun d'eux.

2.2 Fonctionnement d'une extension

En résumé, les fonctionnalités de notre plateforme sont les suivantes :
- Lister les extensions implémentant une interface donnée.
- Lister les extensions implémentant une interface donnée et certaines propriétés.
- Lancer une extension.
- Gérer des plugins classés comme autoruns.
- Gérer des plugins classés comme singleton ou non.

Chapitre 3 : Document pour développeur de plugins

Commencez par importer le projet dans *eclipse* en «forkant» le répertoire github https://github.com/cara-puce/JARVIS.

Pour créer une nouvelle extension, suivez pas à pas les 3 étapes suivantes :

3.1 Créer une classe pour votre plugin

Vous devez d'abord créer une classe implémentant l'interface correspondant au besoin auquel il répond. Vous disposez de 4 interfaces :

- *IAutorun* : une extension autonome appelée directement au lancement de la plateforme.
- *ICreator* : créer un ou plusieurs évènements dans l'agenda.
- *IPrinter* : affiche le contenu de l'agenda dans un objet graphique JPanel.
- *IModifier* : modifie un évènement dans l'agenda.

3.2 Créer le fichier de configuration de votre plugin

Une fois cette classe créée, vous devez ajouter un fichier *.yaml* du même nom (*e.g NewPlugin.yaml*) dans le dossier *pluginConf/*. Le fichier doit avoir le format suivant :

```yaml
name: NewPlugin # Nom du plugin
verbose: Do something # Action effectuée par le plugin
about: Do something amazingly wonderful # Description du plugin
class: plugins.newCreator.NewCreator # Classe implémentant le plugin
interface: platform.plugins.ICreator # Interface du plugin
autorun: "False"
singleton: "False"
```

À noter que pour les extensions implémentant l'interface *IPrinter*, il faut rajouter une clé *default* qui indique si l'extension est l'affichage par défaut ou non. S'il y a plusieurs afficheurs par défaut, le premier trouvé sera utilisé.

3.3 Modifier le fichier de configuration général

Il faut ensuite ajouter l'extension à la liste des extensions gérées par la plateforme dans le fichier *config.yaml* décrit dans la section précédente :

```yaml
...
plugins: # Liste des plugins gérés par la plateforme.
    - ...
```

```
4     - plugins.newCreator.NewCreator
```

Vous venez de créer une extension pour JARVIS, félicitations !

Vous pouvez appeler votre extension en cliquant sur le bouton portant le nom que vous avez renseigné dans la clé *verbose* du fichier de configuration de l'extension.