iPOJO Builder Eclipse Plugin Documentation

Version alpha

Thomas Calmant, Olivier Gattaz

03 February 2011

Table des matières

Contenu :

CHAPITRE 1

Références de développement Eclipse

1.1 Références de développement

1.1.1 Intéractions avec le workspace

Modifications de ressources :
– Suivi des modifications de ressources
– Traitement des modifications de ressources
– Hook sur les modifications de ressources
– Ressource “dérivées”
Gestion de projet :
– Nature de projet
Builder :
– Builder incrémental

1.1.2 Intéractions avec JDT

Quelques références pouvant être utiles :
– Exécution d’un programme Java

1.1.3 Builder Eclipse

Nature de projet

Un builder est associé à un projet en passant par la notion de *nature*. Cette notion permet également d’associer un projet à des plugins.

L’association d’un builder à un projet se fait à l’aide de la méthode *configure()*.

Point d’extension	org.eclipse.core.resources.natures
Interface	IProjectNature
Références	Page 123-136 du diaporama de M. Baron ;

Builder

Le compilateur en lui-même. Il existe un modèle fourni par Eclipse sur lequel on peut s'inspirer.

Point d'extension	org.eclipse.core.resources.builders
Interface	IncrementalProjectBuilder (classe)
Références	Page 108-122 du diaporama de M. Baron ;

1.1.4 Éditeur de fichier metadata.xml

Voir les références utilisées pour le plugin **ReST Editor**

1.1.5 Liens utiles pour iPOJO

– Principe de la manipulation

CHAPITRE 2

Document développeur

2.1 Description du plugin

Ce projet est un plugin pour Eclipse Helios permettant d'utiliser facilement iPOJO dans cet environnement de développement. Il fournit une nouvelle nature de projet et un "builder" associé à cette nature.

Le but n'est pas (actuellement) d'assister l'utilisateur dans la création des fichiers de description iPOJO (metadata.xml, annotations, ...) mais de lui permettre d'utiliser des bundles iPOJO dans ses configurations d'exécution sans avoir à passer par Maven ou par un fichier JAR à placer dans une "Target Platform" de test.

2.2 Outils existants

Nous avons trouvé deux outils concernant l'intégration d'iPOJO dans Eclipse :
- Le plugin Eclipse fourni par Apache Felix Site Web : http ://felix.apache.org/site/ipojo-eclipse-plug-in.html Il permet d'exporter le projet sous forme d'un JAR traité par iPOJO. Géré par Clément ESCOFFIER, il ne semble pas avoir évolué depuis 2008.
- Le builder iPOJO du projet CADSE du laboratoire ADELE (Grenoble) Site Web : http ://code.google.com/a/eclipselabs.org/p/cadse/ Découvert sur le tard, il s'agit d'un projet équivalent au notre. Géré par Stéphane CHOMAT, la dernière mise à jour date de Septembre 2010.

2.3 Principe de fonctionnement

Le principe du plugin est d'ajouter la nature "iPOJO" à un projet existant, ajoutant notre builder à sa liste. Ce plugin ajoute également une entrée "Update Manifest" dans le menu contextuel des fichiers Manifest.mf, permettant de faire une "Pojoization" manuelle.

Lorsque le builder est appelé, ou qu'une mise à jour manuelle est demandée, le plugin demande une compilation complète du projet au plugin JDT, puis utilise l'outil iPOJO Manipulator pour effectuer le traitement des fichiers .class générés et du fichier Manifest. Il ne s'agit pour le moment que d'une recherche des fichiers dans le projet et le format Eclipse pour les transmettre au Manipulator, utilisant les interfaces Java standards.

2.4 Fonctionnalités du plugin

2.4.1 Fonctionnalités attendues

Pojoization automatique des fichiers .class

– Les fichiers class doivent être *pojoizés* avant l'export de fichier JAR ou l'exécution d'une *Run Configuration*.
– Le meilleur moyen d'assurer cette fonctionnalité est d'effectuer le traitement sur les fichiers class fraîchement compilés par JDT.

Builder Eclipse pour la Pojoization

Le plugin doit fournir un "builder" s'insérant dans la chaîne de compilation d'un projet Java utilisant ou non Maven.

Ce builder doit modifier les fichier class et le fichier Manifest.mf du projet dès qu'un de ces fichiers a été modifié par un autre compilateur ou par l'utilisateur.

Le builder peut être soit un plugin Eclipse pur, soit un plugin JDT.

Compatibilité avec le plugin Maven

Le plugin iPOJO doit être capable d'interagir avec le plugin Maven pour Eclipse. Ces deux plugins effectuent des opérations sur les fichiers class dès qu'une modification a eu lieu dans un fichier du projet, ce qui pourrait entraîner une boucle sans fin.

2.4.2 Fonctionnalités optionnelles

Éditeur de fichier metadata.xml

– Au moins fournir un template pour l'éditeur XML, avec les XML schemas renseignés dans le prototype.
– Ajouter la completion des noms java, des noms de composant et de propriétés connus

CHAPITRE 3

Documentation utilisateur

3.1 Description du plugin

Ce projet est un plugin pour Eclipse Helios permettant d'utiliser facilement iPOJO dans cet environnement de développement. Il fournit une nouvelle nature de projet et un "builder" associé à cette nature.

Le but n'est pas (actuellement) d'assister l'utilisateur dans la création des fichiers de description iPOJO (metadata.xml, annotations, ...) mais de lui permettre d'utiliser des bundles iPOJO dans ses configurations d'exécution sans avoir à passer par Maven ou par un fichier JAR à placer dans une "Target Platform" de test.

3.2 Outils existants

Nous avons trouvé deux outils concernant l'intégration d'iPOJO dans Eclipse :

- Le plugin Eclipse fourni par Apache Felix Site Web : http ://felix.apache.org/site/ipojo-eclipse-plug-in.html Il permet d'exporter le projet sous forme d'un JAR traité par iPOJO. Géré par Clément ESCOFFIER, il ne semble pas avoir évolué depuis 2008.
- Le builder iPOJO du projet CADSE du laboratoire ADELE (Grenoble) Site Web : http ://code.google.com/a/eclipselabs.org/p/cadse/ Découvert sur le tard, il s'agit d'un projet équivalent au notre. Géré par Stéphane CHOMAT, la dernière mise à jour date de Septembre 2010.

3.3 Principe de fonctionnement

Le principe du plugin est d'ajouter la nature "iPOJO" à un projet existant, ajoutant notre builder à sa liste. Ce plugin ajoute également une entrée "Update Manifest" dans le menu contextuel des fichiers Manifest.mf, permettant de faire une "Pojoization" manuelle.

Lorsque le builder est appelé, ou qu'une mise à jour manuelle est demandée, le plugin demande une compilation complète du projet au plugin JDT, puis utilise l'outil iPOJO Manipulator pour effectuer le traitement des fichiers .class générés et du fichier Manifest. Il ne s'agit pour le moment que d'une recherche des fichiers dans le projet et le format Eclipse pour les transmettre au Manipulator, utilisant les interfaces Java standards.

CHAPITRE 4

Index et tables

- `content/glossaire`
- *genindex*
- *search*