

CREER ET TESTER LE PLUGIN HELLO WORLD

OBJET :	Documentation technique
RÉFÉRENCE :	Helloworld , hello world, plugin eclipse, plugin
ÉQUIPE :	Méthode
AUTEUR(S) :	POITEVINEAU ROMAIN
DESTINATAIRE(S) :	Développeurs
VERSION / DATE :	1.0, le 11/01/2012
STATUT :	RELEASE CANDIDATE
VALIDÉ PAR :	Développeurs

Remarque

Ce document permet de créer et tester un premier plugin type « Hello World ! » Sur Eclipse.

En cas de problème rencontrer ne figurant pas dans la documentation merci de transmettre le problème à l'équipe méthode. Merci également d'en faire parvenir la solution si vous l'avez trouvé.

Sommaire

Remarque.....	1
Sommaire.....	2
Introduction.....	3
Création d'un projet de PLUG-IN.....	3
Tester un PLUG-IN.....	7
Historique des versions.....	8

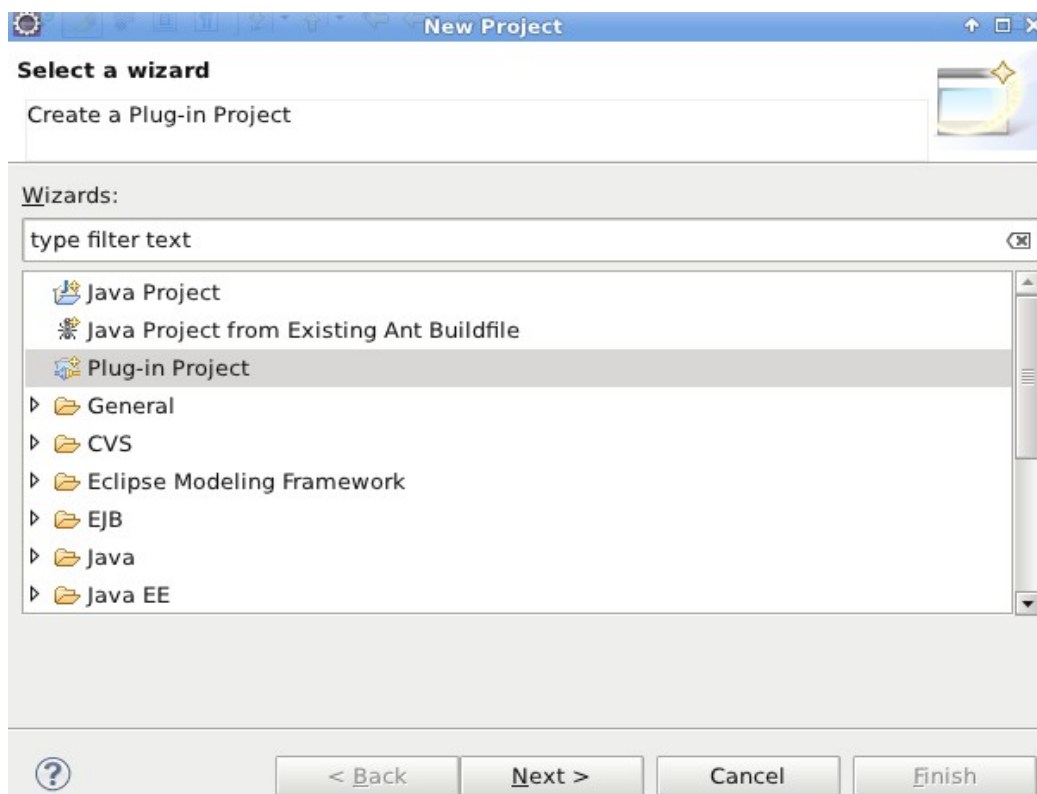
Introduction

Pour simplifier le développement de plug-ins, l'environnement de développement Eclipse (Eclipse SDK) intègre un outillage spécifique : le **PDE** (Plug-ins Development Environment). Basé sur l'outillage Java, le PDE complète l'environnement de développement Java avec des outils prenant en compte les étapes spécifiques au développement de plug-ins (édition des fichiers manifestes, lancement d'un environnement de test, ...).

Création d'un projet de PLUG-IN

La première notion proposée par le PDE est un type de projet particulier : '**Plug-in Project**'. La création d'un plug-in se fait donc en utilisant l'assistant de création d'un 'Plug-in Project' :

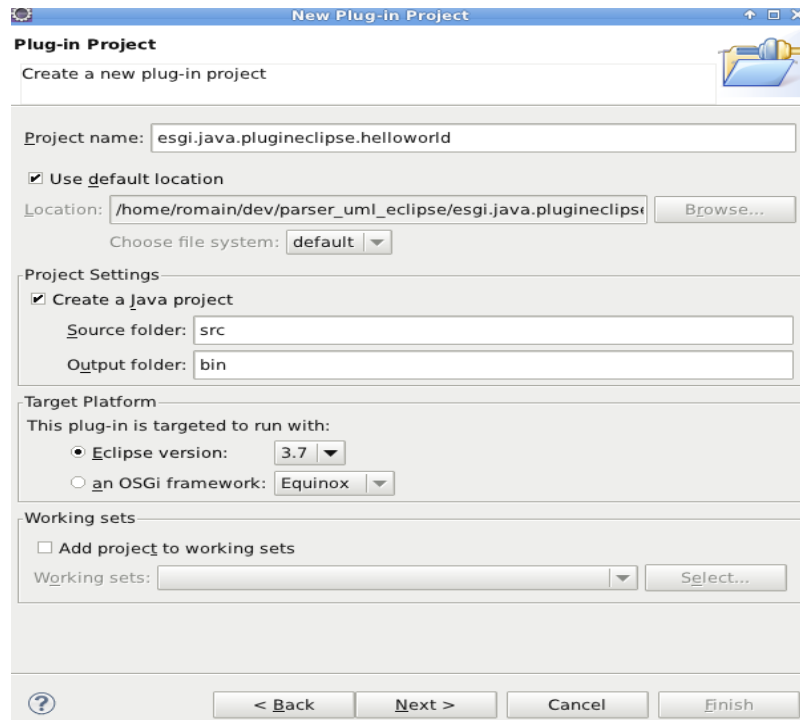
Clicker sur : [File] → [New] → [Project]



CREER ET TESTER LE PLUGIN HELLO WORLD UML ARCHITECT

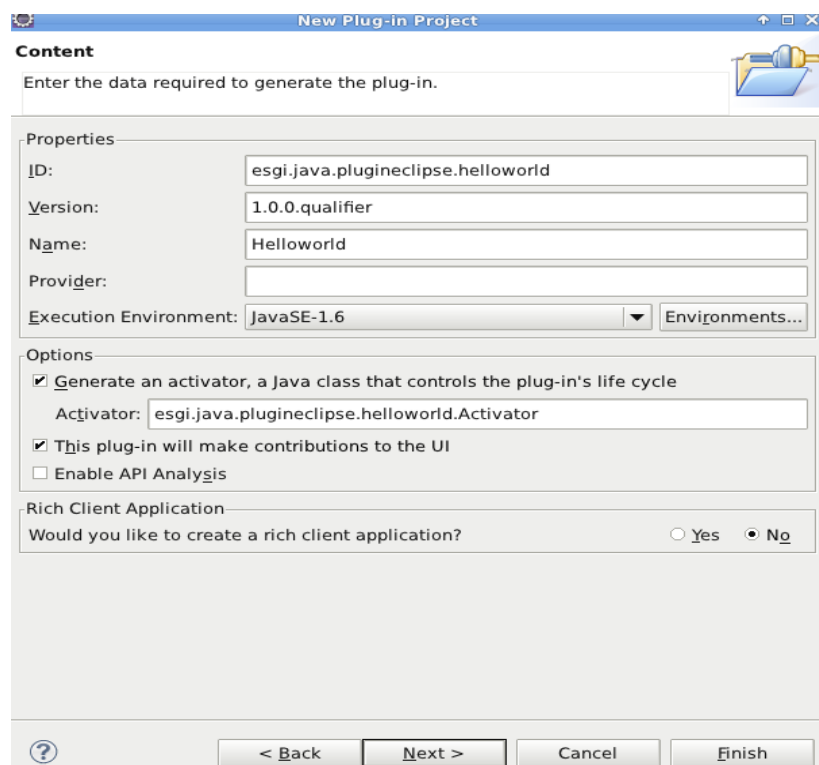
Dans la première page de l'assistant, l'information importante est le nom du projet.

Pour ce nom la convention est d'utiliser les mêmes règles de nommage que pour les packages :



The screenshot shows the 'New Plug-in Project' dialog box. The 'Project name' field is filled with 'esgi.java.pluginclipse.helloworld'. The 'Use default location' checkbox is checked. The 'Location' field shows the path '/home/romain/dev/parser_uml_eclipse/esgi.java.pluginclipse'. The 'Choose file system' dropdown is set to 'default'. Under 'Project Settings', the 'Create a Java project' checkbox is checked, with 'Source folder' set to 'src' and 'Output folder' set to 'bin'. Under 'Target Platform', 'Eclipse version' is selected with a value of '3.7'. Under 'Working sets', the 'Add project to working sets' checkbox is unchecked. At the bottom, there are buttons for '< Back', 'Next >', 'Cancel', and 'Finish'.

Pour la seconde page de l'assistant, les valeurs par défaut sont généralement acceptables :



The screenshot shows the second page of the 'New Plug-in Project' dialog box, titled 'Content'. It prompts the user to 'Enter the data required to generate the plug-in.' The 'Properties' section contains fields for 'ID' (filled with 'esgi.java.pluginclipse.helloworld'), 'Version' (filled with '1.0.0.qualifier'), 'Name' (filled with 'Helloworld'), 'Provider' (empty), and 'Execution Environment' (set to 'JavaSE-1.6'). The 'Options' section has three checkboxes: 'Generate an activator, a Java class that controls the plug-in's life cycle' (checked), 'This plug-in will make contributions to the UI' (checked), and 'Enable API Analysis' (unchecked). The 'Activator' field is filled with 'esgi.java.pluginclipse.helloworld.Activator'. The 'Rich Client Application' section asks 'Would you like to create a rich client application?' with 'Yes' and 'No' radio buttons, where 'No' is selected. At the bottom, there are buttons for '?', '< Back', 'Next >', 'Cancel', and 'Finish'.

- l'**ID** du plug-in reprend le nom du projet, cette valeur est importante car elle est notamment utilisée par certaines API.
- **la version** peut être librement modifiée.
- **le nom du plug-in**, n'est pas une information vraiment importante, mais elle peut être parfois visible par l'utilisateur (Menu Help->About->bouton 'Plug-ins details').
- même remarque pour **le nom du fournisseur**.
- le champ '**Classpath**' est vide par défaut (depuis Eclipse 3.2). Nous étudierons plus tard la modification de cette valeur.
- La génération d'un '**Activator**' est souhaitable, cette classe respecte une convention proposée par OSGi, elle permettra notamment de réagir à des événements liés au cycle de vie du plug-in (arrêt, démarrage, ...).
- Le fait d'indiquer que le plug-in contient une partie graphique ('**This plug-in will make contributions to the UI**') permet à l'assistant d'ajouter automatiquement les librairies graphiques d'Eclipse dans les dépendances du projet et aussi de choisir la classe dont héritera l'Activator ('**org.eclipse.ui.plugin.AbstractUIPlugin**' pour un plug-in graphique, '**org.eclipse.core.runtime.Plugin**' sinon).
- Le dernier choix de cette assistant permet d'indiquer si ce plugin contiendra le point d'entrée d'une **application Eclipse RCP**. Ce choix n'a aucun impact sur la possibilité de réutilisation du plug-in dans une application Eclipse RCP : tous les plug-ins sont utilisables dans une application Eclipse RCP. Nous étudierons dans un prochain tutorial la construction d'une application Eclipse RCP, l'utilisation de ce même assistant permettra de demander la génération des classes servant de point d'entrée à l'application, le reste de l'application sera constitué de plug-ins standards.

La troisième page de l'assistant offre la possibilité de choisir à partir de quel modèle le projet sera créé, pour ce premier exemple nous utiliserons le modèle '**Hello, World**'. Ce modèle permet de générer un plug-in qui ajoute un menu dans la barre de menu principale avec une entrée qui déclenchera l'ouverture d'une boîte de dialogue affichant un message.

Clicker ensuite sur [Next].

New Hello World plug-in project

Sample Action Set

This template will generate a sample action set extension with a menu, a menu item and a tool bar button.

Java Package Name:

Action Class Name:

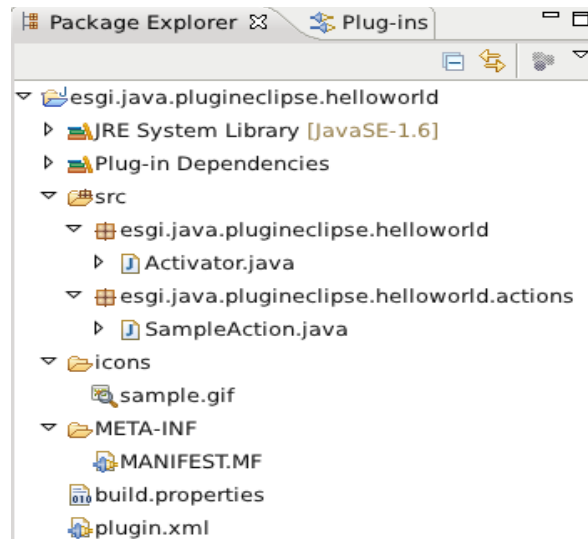
Message Box Text:

? < Back Next > Cancel Finish

La page suivante dépend du modèle choisi précédemment, elle permet de renseigner des informations utilisées pour générer le code du projet. Dans notre cas, utilisation du modèle 'Hello, World', les informations modifiables sont le noms du package, le nom de classe à déclencher lorsque le menu est sélectionné ainsi que le texte du message à afficher

Une fois le bouton '**Finish**' sélectionné, Eclipse propose d'ouvrir la perspective PDE (celle-ci est très ressemblante à la perspective Java) et ouvre l'éditeur des fichiers manifestes dans la zone d'édition.

La structure du projet généré est la suivante :



Tester un PLUG-IN

Les plug-ins obéissent à un cycle de vie particulier. Notamment, les fichiers manifestes sont analysés pendant la phase de démarrage d'Eclipse. Pour que ce cycle de vie soit correctement respecté lors des tests, le PDE propose de lancer une deuxième instance d'Eclipse à partir de celle servant au développement des plug-ins.

Pour lancer le plugin.

Click droit à la racine du projet → [Debug As] → [Eclipse Application].

Une nouvelle instance d'éclipse qui contient le plugin est alors lancé.

Clicker sur le petit bouton eclipse qui est apparus dans la barre d'outils.

Une MessageBox apparaît alors .



Historique des versions

VERSION	DATE	AUTEUR	DESCRIPTION	STATUT
1.0	11/01/2012	POITEVINEAU ROMAIN	Documentation sur les plugins, création et test d'un plugin hello world	TOCHECK