Laboratoire d'Analyse et d'Architecture des Systèmes

LAAS-CNRS

ECLIPSE

Une courte introduction

Résumé:

Quelques notions de bases sont rapidement décrites.

Mots clés:

Eclipse, Version, Exemples, IDE, Plug-ins, Rich Client Platform, Eclipse Modeling Tools, XText

Serge Bachmann

Sommaire

Informations Générale

Fichier source document:	E:\FORMATION_Eclipse\DOC_Générale\IntroductionEclipse.odt
(Usage interne)	
Date de création:	17 septembre 2009
Date d'impression:	5 décembre 2011
Plate forme	Windows 32

1 Historique

En Novembre 2001, **IBM** et sept autres compagnies lancent **Eclipse** comme projet « **Open Source** ». Le succès dépasse tous les espoirs des fondateurs. L'ensemble des nombreuses compagnies qui a rejoint **Eclipse** est organisé en une corporation à but non lucratif: « **Eclipse Foundation** ».

Initialement **Eclipse** permet de développer des Environnement de Développement de Logiciel (Integrated Development Environment, IDE) pour de nombreux langages: **C**, **C++**, **Python**, **Ruby**, et bien sur **Java**

Cette plateforme, qui a fait ses preuves, est mise à la disposition des développeurs pour réaliser des outils de développement et toutes sortes d'applications.

Eclipse fournit des éditeurs, des constructeurs, des débogueurs , des outils permettant de gérer le développement en équipes, … .

De nombreux projets développé sur la plateforme **Eclipse** voient le jour.

1.1 Les versions

Tous les ans début juillet apparaît une nouvelle version d'Eclipse. Chaque version porte un nom:

Nom	année	version
Callisto	2006	3.2
Europa	2007	3.3
Ganymede	2008	3.4
Galileo	2009	3.5
Helios	2010	3.6
Indigo	2011	3.7

2 Notions de base

Eclipse n'est pas un logiciel monolithique mais un petit noyau « plug-in loader » entouré de centaines de plug-ins. Un plug-ins peut attendre un service d'un autre plug-in ou fournir un service à d'autres plug-ins.

2.1 Les plug-ins

Le comportement d'un plug-in est défini par son code mais ses dépendances et ses services sont déclarées dans deux fichiers « MANIFEST.MF » et « plugin.xml ».

Au lancement d'Eclipse, le « plug-in loader » analyse les fichiers « MANIFEST.MF » et « plugin.xml ». Pour chaque plug-in il construit une structure contenant cette information. On évite ainsi de charger l'ensemble du code des « plug-in » et donc d' écrouler la plateforme.

Un plug-in est construit par ajout de points d'extensions (Extension Points) qui sont des références à d'autre plug-ins. Le fichier « plugin.xml » mémorise les points d'extensions du plug-in.

2.2 Le « Workspace »

Eclipse affiche et modifie les fichiers du « Workspace ». Le « Workspace » contient les fichiers utilisateur et des fichiers nécessaires à Eclipse.

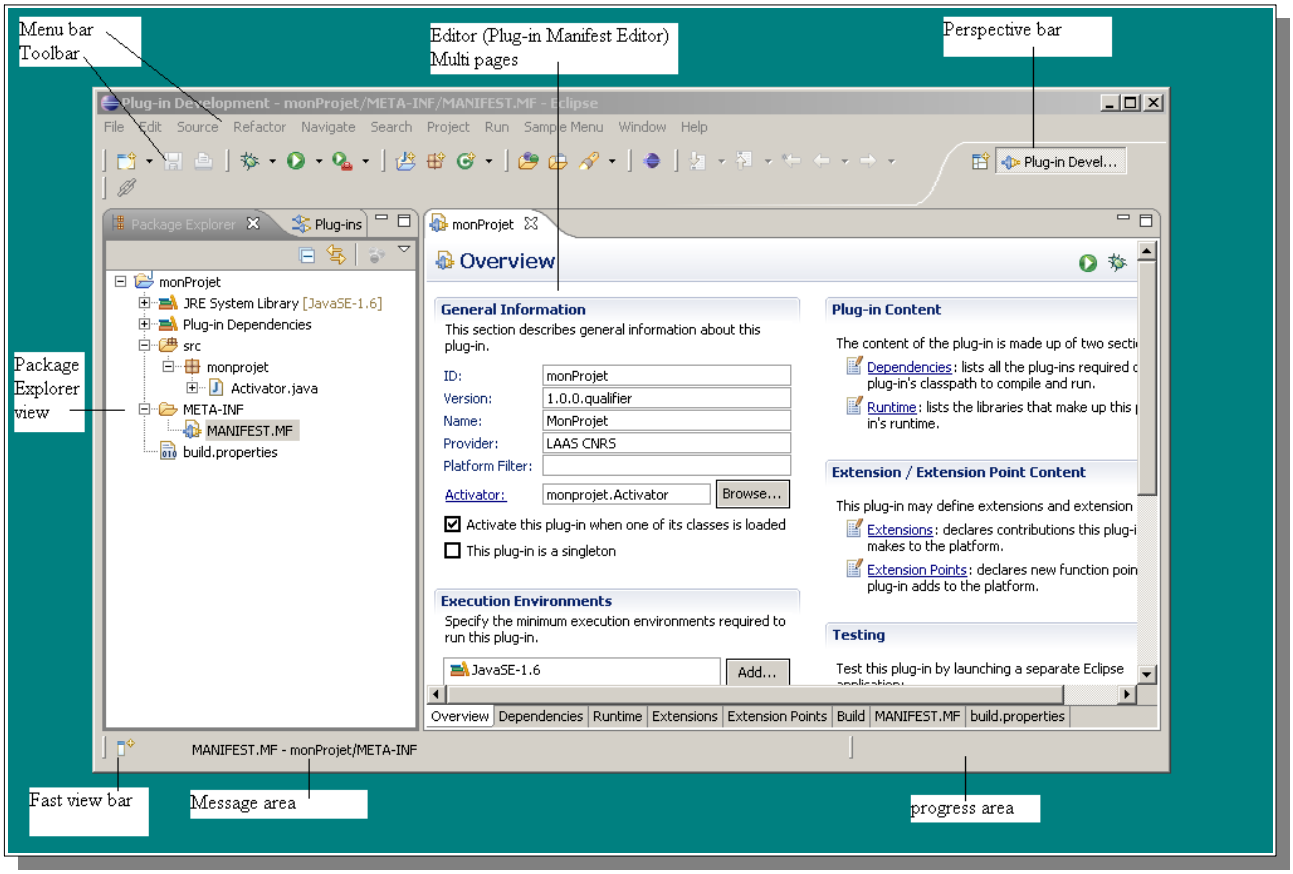
2.3 Le « Workbench »

Le « Workbench » fournit un ensemble de fonctions homogènes :

- pour la création et le « management » des ressources du « Workspace »
- la navigation dans les ressources du « Workspace ».

Lors du lancement d'Eclipse un boite de dialogue permet de préciser l'emplacement du « Workspace ».

Dans le cas du développement d'un « plug-in » l'interface utilisateur à l'aspect suivant:



- Toutes les actions de l'Interface Utilisateur sont accessibles depuis le « **menu bar** »
- Le « **toolbar** » est un ensemble d'icônes qui permet d'accéder à certaines commandes des menus du « **menu bar** »
- Un « **Editor** » permet de visualiser et de modifier les fichiers du « Workspace ».

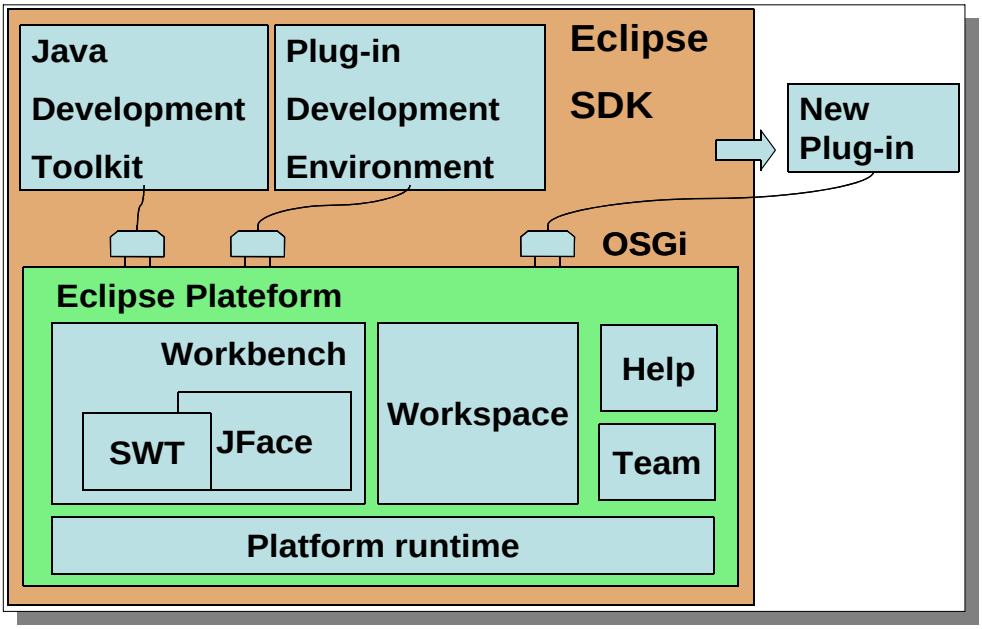
  Un Editeur apparait au centre du « **Workbench** ». Ici l'éditeur est le « **Plug-in Manifest Editor** » qui joue un rôle primordial dans la création des plug-ins.
- Une « **view** » affiche selon un point de vue particulier une information. Ici le « **Package Explorer** » permet de naviguer dans le « **Workspace** »
- Une **perspective** définie et positionne différentes vues autour de l'éditeur.

  La perspective sélectionnée est ici la perspective « Plug-in Development ». Elle définit et positionne l'éditeur et les vues qui constitue l'environnement de développement particulier.
- Pour naviguer dans le projet on utilise le « **Package Explorer view** ».

  Par exemple, un double clic sur le fichier « META-INF / MANIFEST.MF » ouvre le « Plug-in Manifest Editor ».
- La « Fast view bar » permet d'accéder rapidement à une vue.

2.4 La plateforme « Software Development Kit »

La figure ci-dessous décrit le « **Sofware Developement Kit** » (SDK):



Nous allons définir chaque module.

2.4.1 « Eclipse plateform »

La plate-forme Eclipse est organisée en sous systèmes qui sont implémentés sous la forme d'un ou de plusieurs plug-ins. Les sous systèmes sont construits sur la « Plateform Runtime ».

Les sous systèmes:

- **SWT** - « Standard Widget Toolkit » Une librarie de widgets destiné à la construction d'interface utilisateur. Ces widgets sont étroitement intégrés au système d'exploitation.
- **JFace** – Une librairie générale construite au dessus de SWT.
- **Workbench** constitué de:
  - Workbench Core – assure entre autre la gestion des projets
  - Workbench UI – géré les aspects Interface Utilisateur (éditeur, vue, perspectives, actions, préférences)
- **Team** – groupe de plug-ins pour intégrer des outils comme (CVS)
- **Help** - Permet de développer une documentation intégrée à l'Interface Utilisateur.

2.4.2 « Java Developpement Toolkit »

Java Development Toolkit est constitué de:

- JDT core – Plug-ins pour le Java Developpement Tooling (hors UI)
- JDT – plug-in pour Eclipse IDE.

2.4.3 « Plug in development environment »

Environnement de développement de « plug-in » utilisé lors du développement de nos plug-ins.

2.4.4 OSGI

Le mécanismes des plug-in Eclipse est basé sur la notion de « bundles » normalisé par « OSGI Alliance », voir « http://www.osgi.org/.

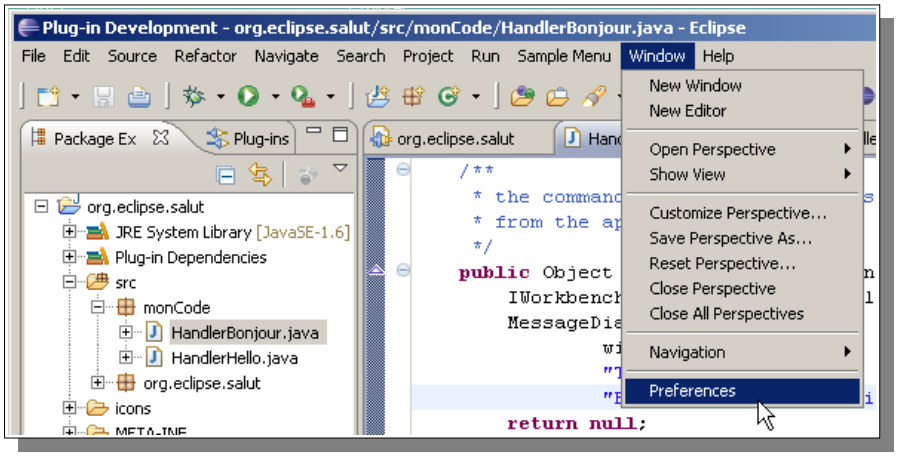
Le terme « Bundle » est parfois utilisé à la place de « plug-in ». Par exemple dans le fichier « MANIFEST.MF » :

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: MonProjet
…
```

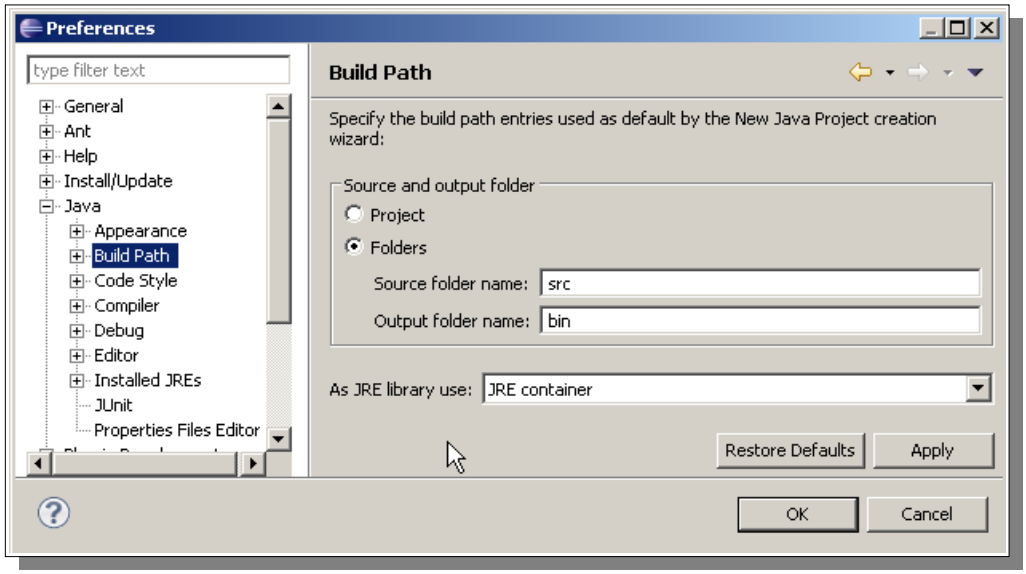
2.5 Les « Preferences »

Afin d'adapter la plateforme Eclipse à nos besoins on dispose d'un ensemble de préférences.

Sélectionner la commande: « Window > Preferences »

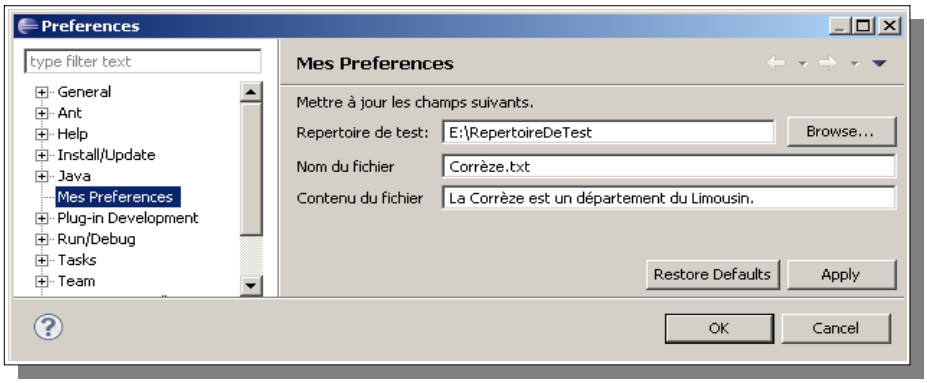


Par exemple les valeurs ci-dessous seront utilisées par le « New Java Project création wizard »



C 'est à dire lors de la création d'un nouveau projet Java.

Il est aussi possible de définir notre propre page de préférences afin d'initialiser notre application:



3 Eclipse packages

Nous avons vu que la principale caractéristique d'éclipse est sa modularité. La fondation Eclipse fournie différents ensembles de plug-in pour répondre à des besoins particuliers, par exemple:

- IDE C++, ...
- Plug-in developers,
- etc...

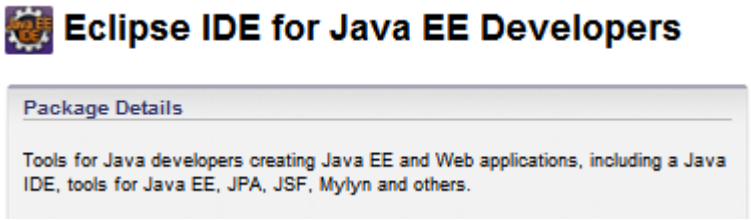
3.1 Packages standards

Pour éviter tout problème d'incompatibilité entre différentes versions de plug-ins vous trouverez des packages tout prêt (http://www.eclipse.org/downloads/), faite le bon choix!

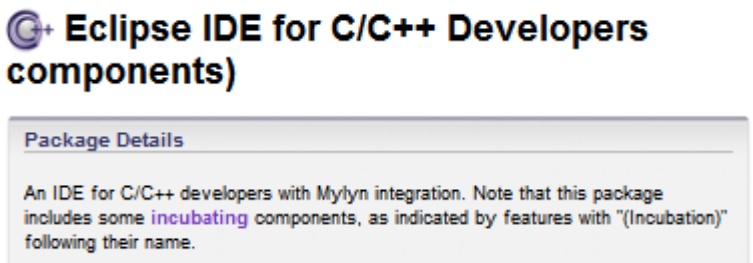
3.1.1 IDE:

Des Environnement de développement intégrés pour développer des applications:

Java:



C/C++:

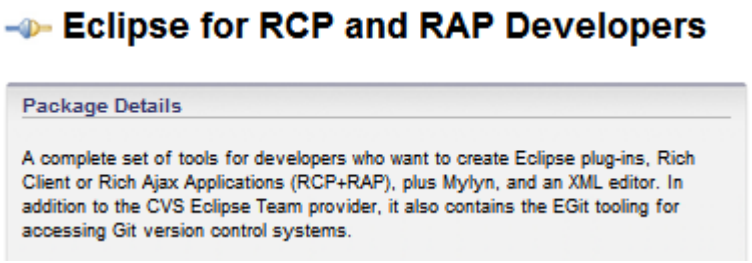


…

3.1.2 Developpement de plug-ins

Pour développer des plug-ins on pourra utiliser:

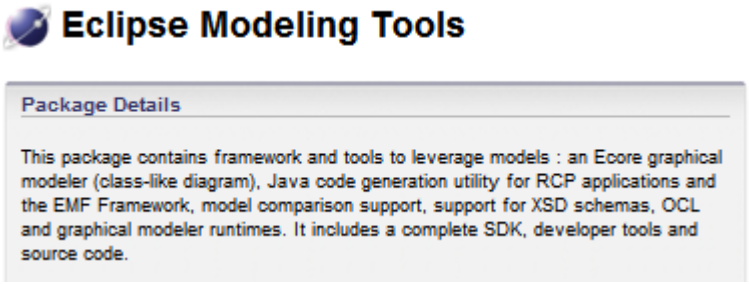
- Développement de plug-ins:



ce « package » permet:

  - de développer de nouveaux plug-ins.
  - de développer des applications Eclipse RCP (Rich Client Platform) qui s'exécutent indépendamment d'une plate-forme Eclipse.

- Développement de plug-ins à partir de méta-modèle:



3.2 Package configuré sur mesure

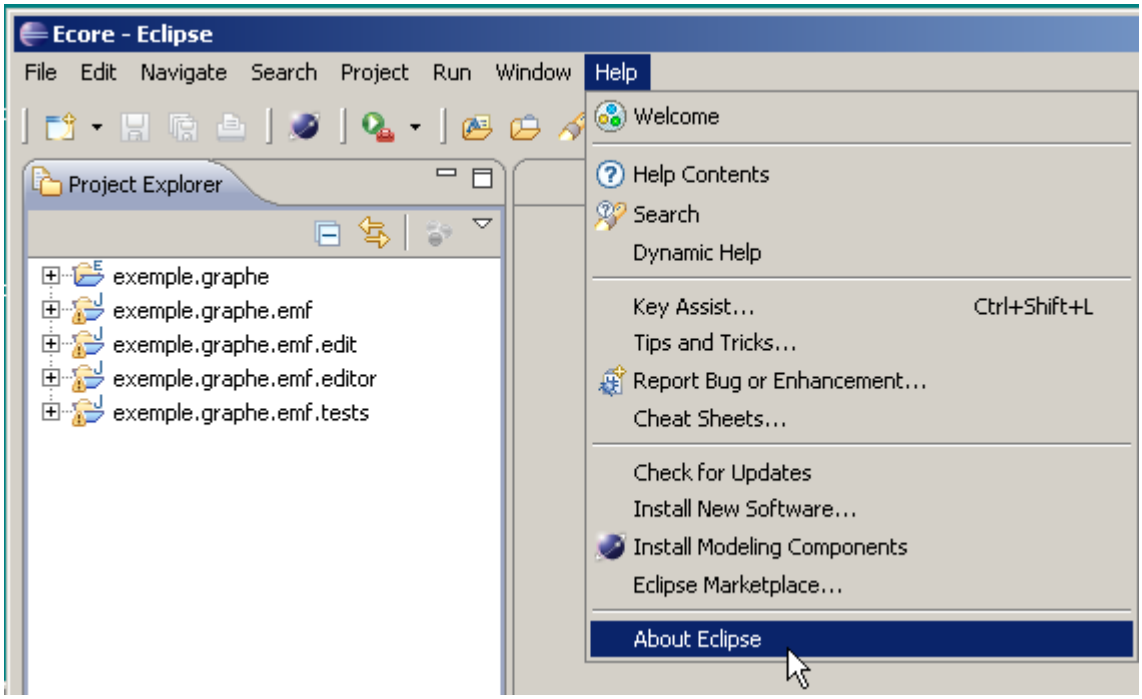
Pour réaliser une configuration sur mesure utilisez le site:

http://ondemand.yoxos.com/geteclipse/start

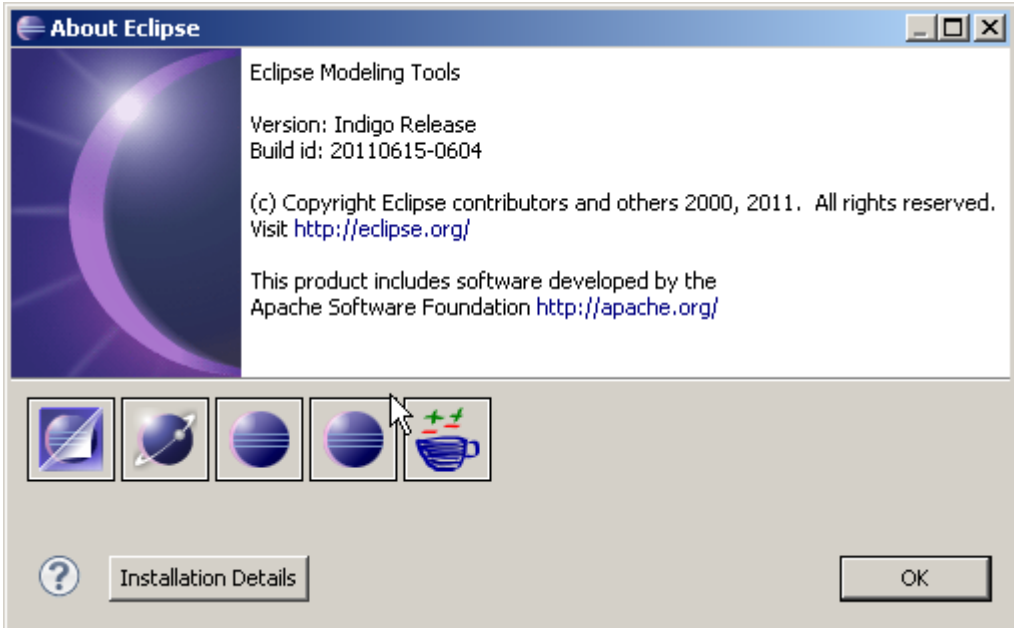
4 Quelques trucs et astuces

4.1 Pour connaître la version d'Eclipse utilisée

Sélectionner la commande « Help > About Eclipse »



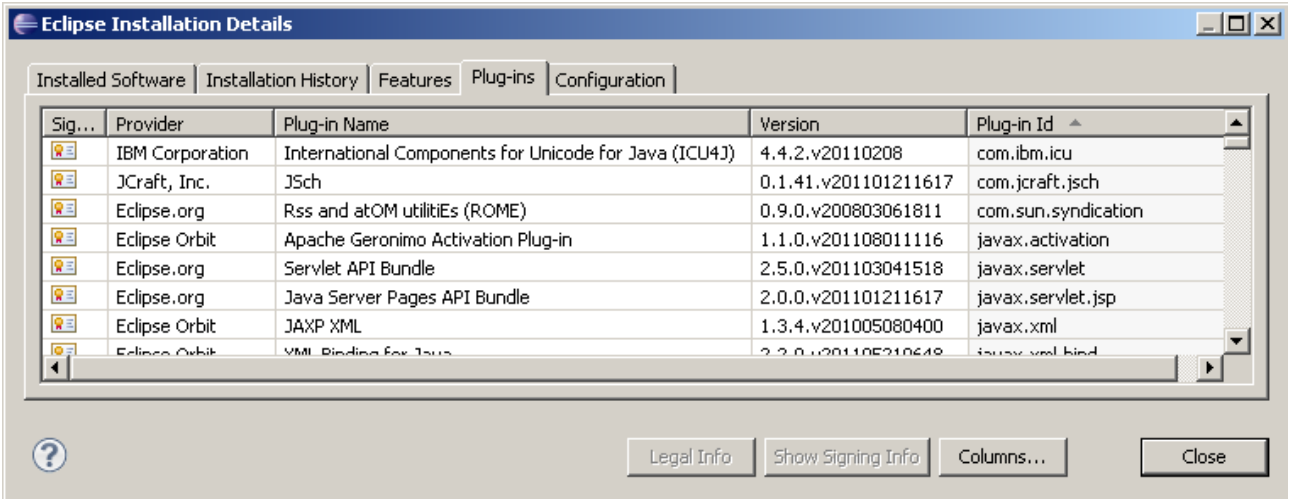
on obtient:



4.2 Pour connaître les plug-ins installés

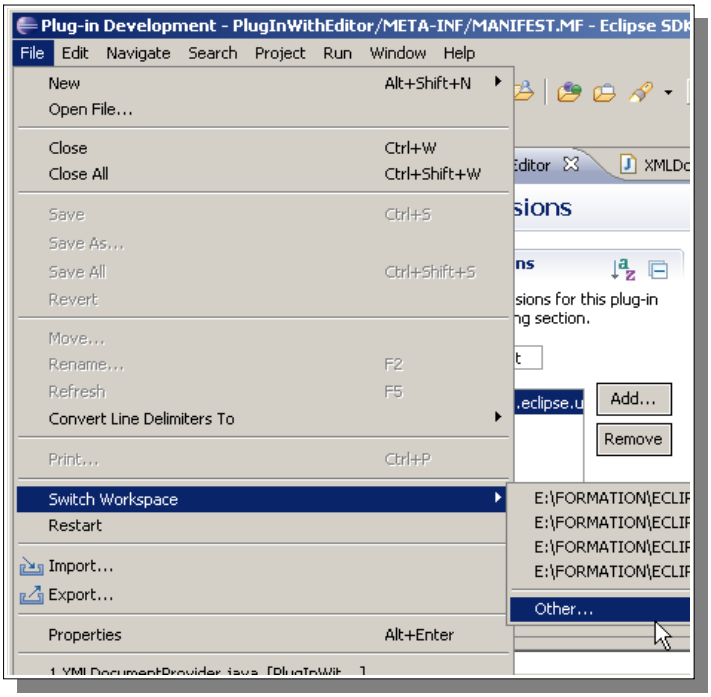
Dans la fenêtre « About Eclipse » (voir paragraphe précédent) cliquer « Installation Details ». Dans « Eclipse Installation Details » cliquer « Plug-ins » pour afficher la liste des plugins.

En cliquant dans « Plug-in Id » on trie les plugins par ordre alphabetique (ou l'ordre inverse):

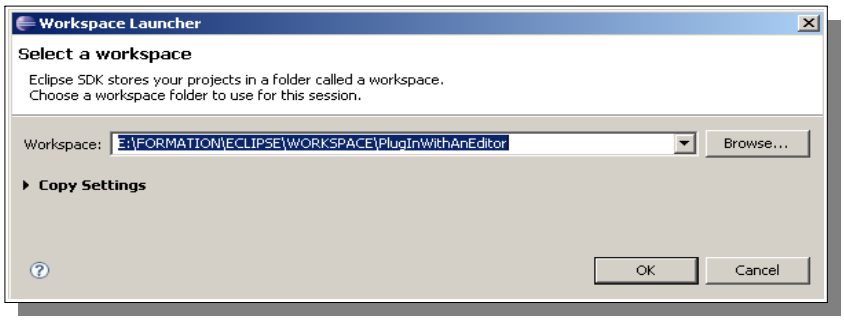


4.3 Pour connaître le « Workspace » en cours d'utilisation

Sélectionner:« File > Switch Workspace > Other... »



Dans « Workspace Launcher – **Select a workspace** » le workspace courant est visualisé:



Cliquer « Cancel »

4.4 Modification directe dans le « Workspace »

Bien que cette pratique soit peu orthodoxe, il peut être commode d'éditer, par exemple, un icône directement dans le « Workspace ».

Moyennant quelques précautions, on peut intervenir directement dans le « Workspace », il est cependant préférable de créer au préalable les fichiers avec les commandes Eclipse.

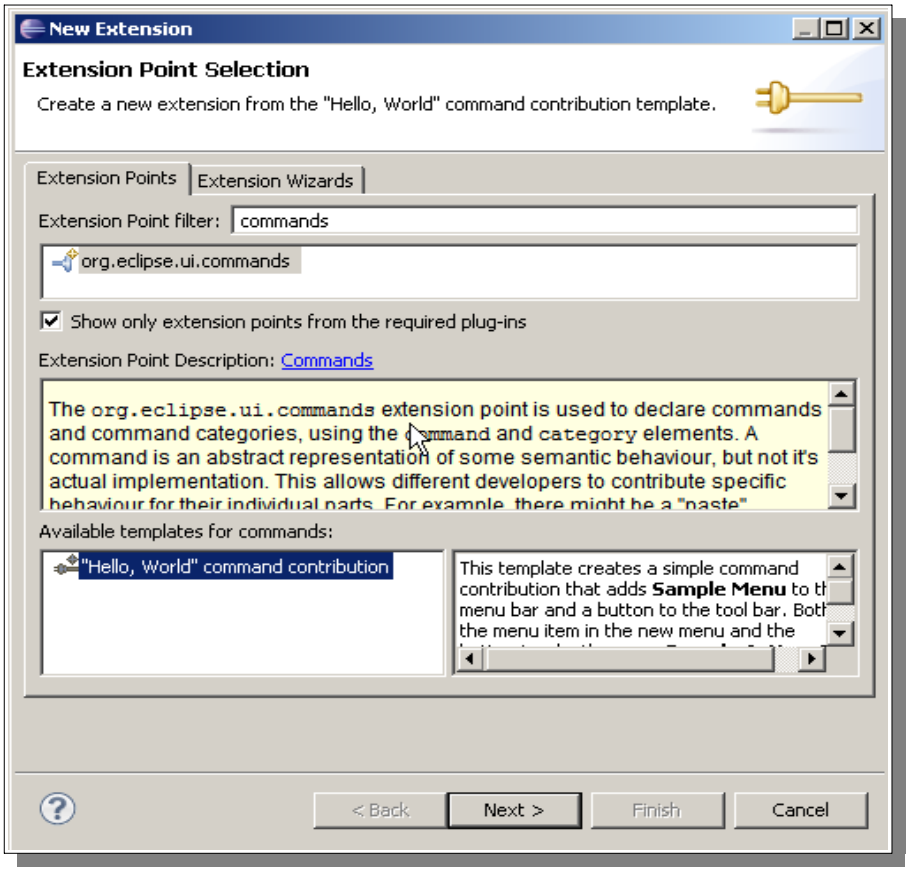
Dans le cas ou il y a divergence entre Eclipse et le contenu du « workspace » un message « Ressource is out of sync with the file system: nom de la ressource » s'affiche.

Faire un clic droit sur la ressource dans le « package explorer » et sélectionner dans le menu contextuel la commande:

5 Documentation

5.1 Documentation sur un point d'extension

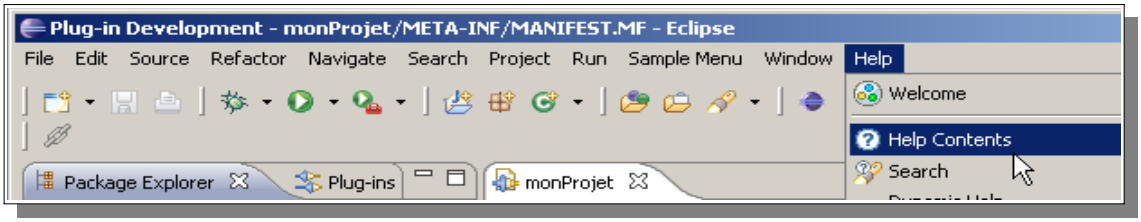
Lors de la création d'un point d'extension dans « New Extension – **Extension Point Selection** » une description du plug-in est donnée. Éventuellement un « template » est fourni.



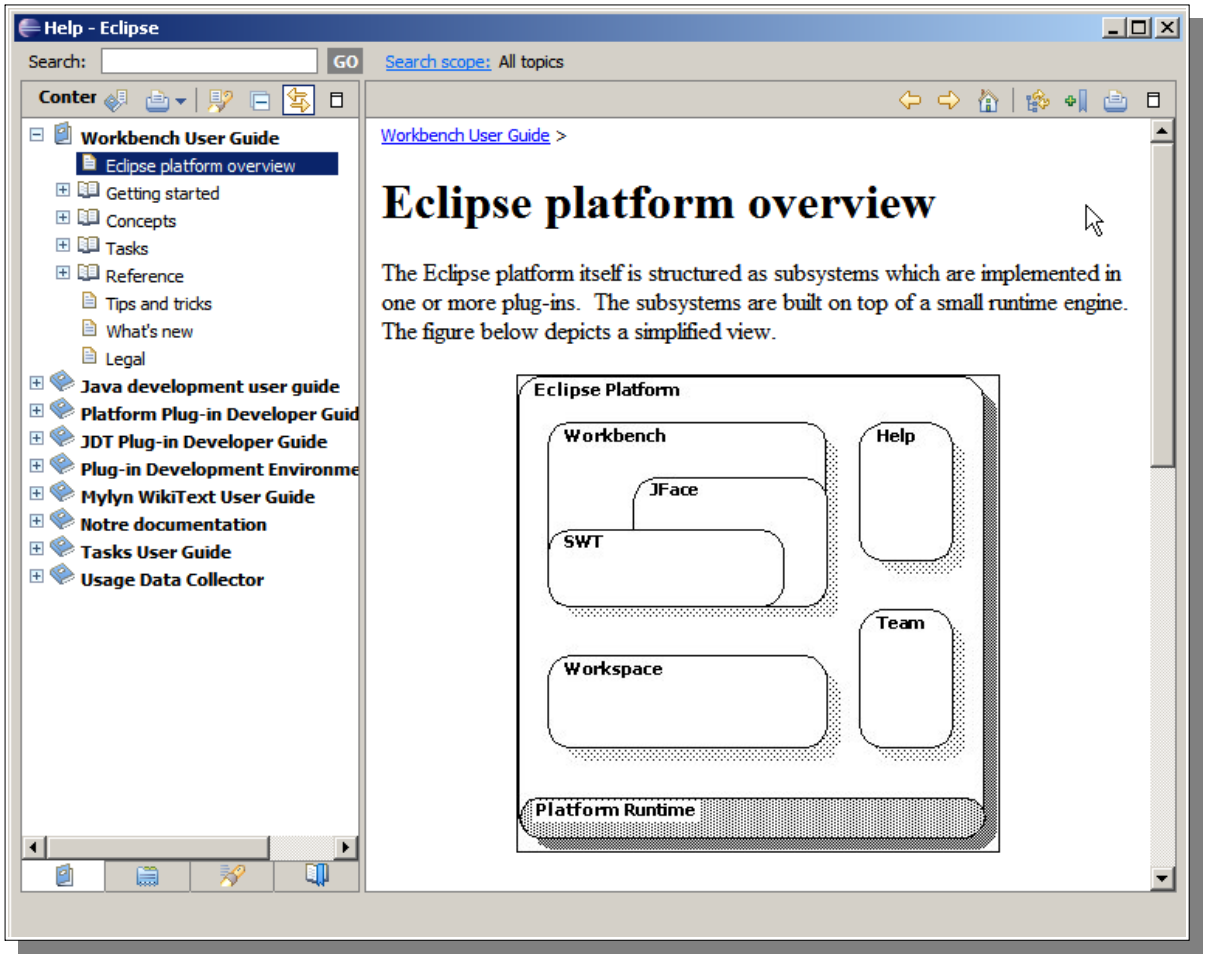
Pour comprendre le fonctionnement d'un point d'extension on peut sélectionner un « template ».

5.2 Documentation en ligne

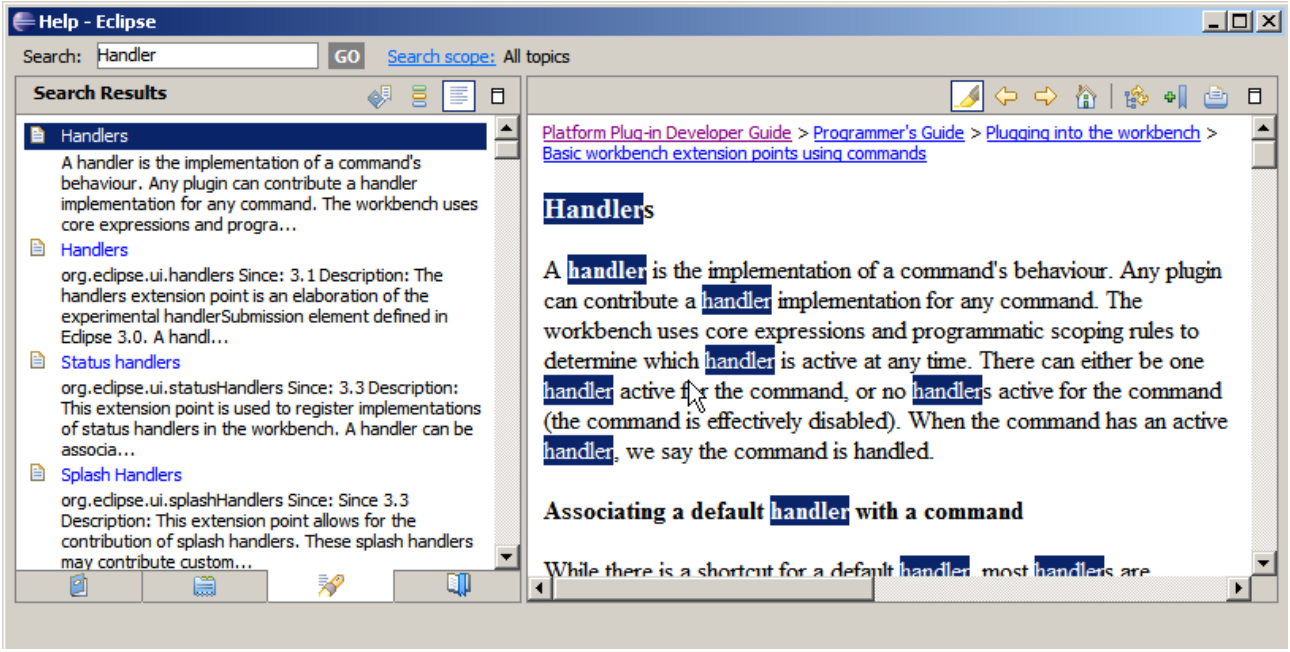
Sélectionner « Help > Help Contents »:



La documentation en ligne s'affiche:

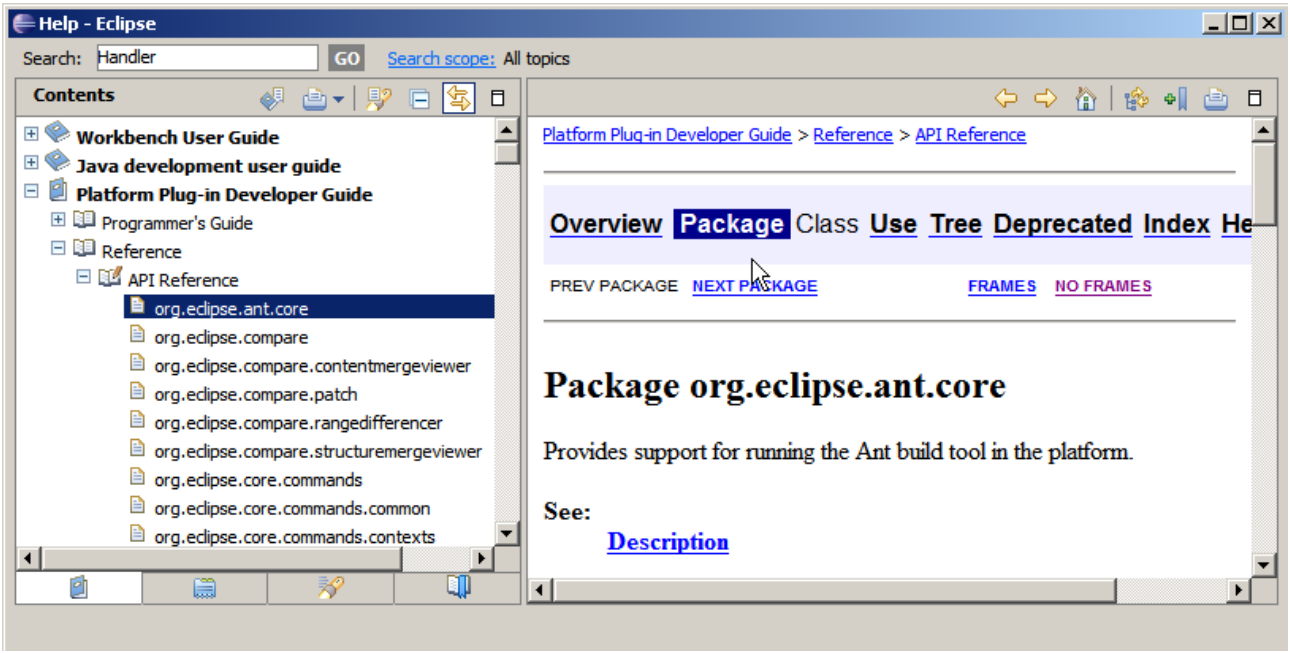


Il est possible de réaliser une recherche:



Ou d'obtenir l'ensemble des « API » dans

« Platform Plug-in Developer Guide > Reference> API Reference »



Remarque: *Il ne faut pas chercher à programmer directement à partir de la définition d'une API mais à partir d'exemples d'utilisation de l'API!*

5.3 D'autres sources

Plusieurs sources sont à votre disposition.

Sous Eclipse en cours de développement

- La documentation sur un point d'extension
- La documentation en ligne
- L'aide fournie par les éditeurs

Hors plateforme Eclipse

- Les sites dont le site officiel:

  http://www.eclipse.org/

- The Eclipse Series informit.com/series/eclipse
  eclipse Plug-ins (third Edition)
  eclipse Modeling Framework (Second Edition)
  eclipse Rich Client Platform (Second Edition)
  eclipse Modeling Project

Cours de « Lars Vogel »

- http://www.vogella.de/eclipse.html

6 Différentes utilisation d'Eclipse

Il existe bien des façons d'utiliser Eclipse. Dans l'ensemble des documents proposés nous allons aborder, à travers des exemples, les points suivants :

- L'utilisation de l'environnement de développement intégré Java,
- Le développement de plug-ins, les notions de base,
- Le développement d'applications indépendantes,
- La génération de plug-ins à partir de Méta-modèles,
- La génération d'éditeurs syntaxiques coloré adaptés à un langage spécifique à un domaine.

7 Conclusions

Quelques notions de bases on étaient rapidement exposées. Le lecteur se reportera aux différentes sources citées pour compléter ses connaissances.

8 Licence

La licence « créative commons » : http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

s'applique à ce document.