

# Eclipse en 1010 minutes



CREATION : 2009/11/2  
 MISE A JOUR : 2011/11/29  
 VERSION d'ECLIPSE : GANYMEDE/GALILEO/INDIGO

Utiliser un EDI est impératif lorsque l'on veut travailler de manière conviviale et rapide. Eclipse fait partie des outils incontournables pour développer en Java. Sa grande force ? il est hautement extensible même si nous n'allons pas utiliser cette propriété. On peut même, en installant les plugins idoines, l'utiliser pour d'autres langages de programmation (CDT pour C++ ou encore PDT pour PHP).

Ce n'est pas le seul outil de sa catégorie : **NetBeans** mis en avant par Sun/Oracle est également très intéressant d'autant plus que celui-ci intègre les toutes dernières JSR. J'oublie également **Intelli J** pour une question de licence.

Ce petit tutorial a été réalisé grâce à l'aide intégrée à Eclipse qui est plutôt bien faite. Je vous conseille notamment d'étudier les raccourcis clavier qui sont très riches.

## Première utilisation : workspace, projet, perspective et vues

A la première utilisation, Eclipse vous demande quel est le **workspace**, c'est-à-dire le répertoire maître où le programme va placer toutes vos œuvres et les informations sur celles-ci.

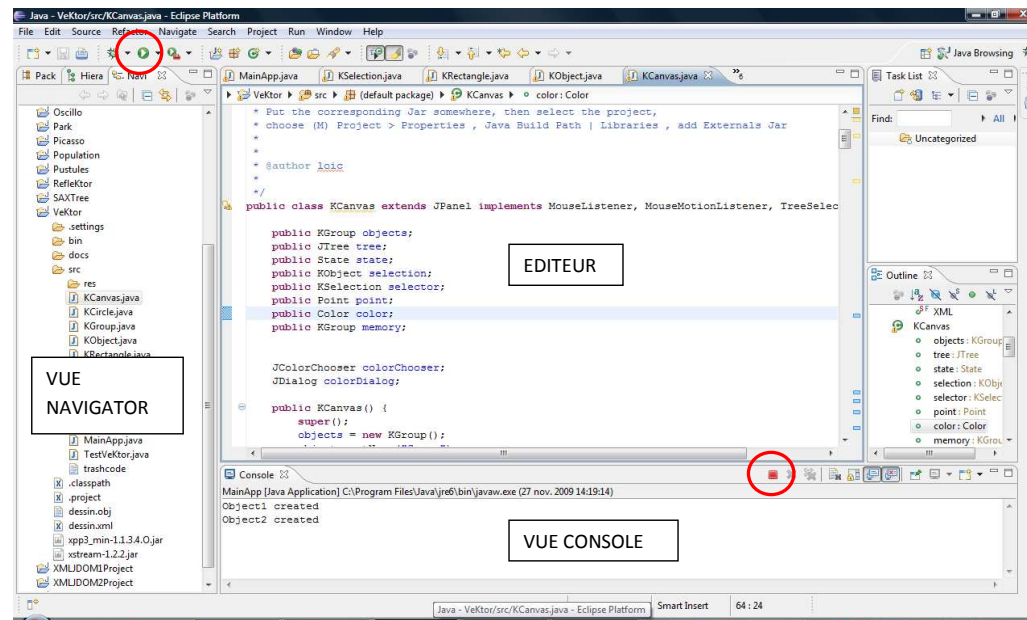
Ce **workspace** contient les différents projets sur lesquels vous travaillez. Il est, en effet, très fortement conseillé de travailler par **projet**. Pour chaque nouveau programme que vous voulez créer, vous devez lui associer un projet : un répertoire spécifique qui contiendra vos fichiers sources, vos fichiers compilés, vos bibliothèques, vos documentations et toutes les ressources que vous estimez nécessaires.

Lorsque vous travaillez sur un code, vous le faites dans un but précis, ce qu'Eclipse appelle une **perspective**, qu'il est possible de paramétrer. On peut citer par exemple :

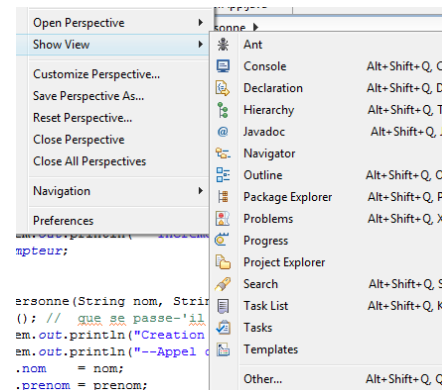
- **Java** pour créer son code et le compiler
- **Debug** pour déboguer votre programme
- **Java browsing** pour parcourir facilement la hiérarchie des projets

Le nombre de perspectives disponibles dépend de la version d'Eclipse installée et des différents plugins.

Voici un exemple de perspective **Java** :



La perspective est en général constituée de l'éditeur java et de différentes **vues**.



Vues intéressantes :

- **Navigator** pour lister les différents projets
- **Console** pour suivre l'exécution de programme. Il y a notamment un bouton rouge d'arrêt du programme.

Si vous programmez en Swing, attention à ce que le programme courant soit bien arrêté et pas seulement non visible car cela consomme des ressources. L'exécution des programmes sous Eclipse se fait dans Eclipse même. Assez vite, on obtient un message comme quoi il n'est plus possible d'exécuter des programmes (il faut alors comprendre que la limite de création de nouveaux threads est atteinte). Fermez les programmes en cours (bouton rouge) et tout rentrera dans l'ordre.

Assez vite, on obtient un message comme quoi il n'est plus possible d'exécuter des programmes (il faut alors comprendre que la limite de création de nouveaux threads est atteinte). Fermez les programmes en cours (bouton rouge) et tout rentrera dans l'ordre.



Annotations in the screenshot:

- COMMENTAIRE JAVADOC (points to a Javadoc comment)
- WARNING (points to a warning icon in the toolbar)
- ERREUR (points to an error icon in the toolbar)
- WARNING ET SUGGESTION (points to a warning and suggestion icon in the toolbar)
- ERREUR ET SUGGESTION (points to an error and suggestion icon in the toolbar)
- TODO (points to a TODO icon in the toolbar)
- BARRE VERTICALE // PROGRAMME ENTIER (points to the vertical toolbar)

L'éditeur est de loin le plus intéressant, voici quelques propriétés. La liste n'est pas exhaustive.

- Compilation automatique en arrière-plan ou forcée par une sauvegarde. La compilation automatique se désactive (cf la FAQ)
- Exécution standard ou en mode débogage (bouton lecture ou insecte)
- Affichage des erreurs de compilations, des warnings ou des TODO en deux endroits : là où la « situation » a été détectée, l'autre dans la barre sur le côté droit qui représente le fichier dans son intégralité.
- Proposition de résolution pour les warnings et les erreurs
- Edition du code intelligente : import des classes, indentation automatique ou reformatage pour se conformer aux conventions de nommage de Sun/Oracle
- Redéfinition des méthodes ou implémentation à vide des méthodes définies dans une interface
- Adapté au développement agile et au refactoring



## A venir : la perspective de débogage...

Même si en Java, il n'y a pas de pointeur, il est tout à fait possible d'avoir des comportements non désirés. Si vous ne voulez pas déboguer à coup de `println`, et surtout si vous voulez **être efficace**, vous devez prendre l'habitude d'utiliser un débogueur et cela tombe bien, Eclipse s'interface avec celui de Sun/Oracle. Il est disponible au travers de la perspective **Debug**.

Lorsque l'on va pas à pas, il est tout à fait possible de « sauter » dans une méthode d'une classe de la plateforme Java. Pour ne pas avoir de message d'erreur lié à l'absence du **code source**, il faut l'installer et le télécharger ;-). C'est super facile, il suffit de choisir la bonne version du kit de développement sur la page de Sun.



## Le projet

Un projet est l'unité d'organisation de votre programme et il est lui-même organisé en répertoires. Je vais donner une liste des plus courants.

Un répertoire **src** qui contient tous vos codes sources éventuellement organisés en **packages**.

Un répertoire **bin** qui contient le pseudo code. **ATTENTION**, ce répertoire n'est pas permanent et est régulièrement réécrit/vidé par Eclipse. En fait, ce répertoire contient une copie de **src** où tous les codes sources sont compilés. Pour synchroniser les répertoires **src** et **bin**, il suffit de le faire par la touche F5 (ou Refresh par le menu contextuel).

Un répertoire **doc** qui contient la documentation de votre application au format **javadoc**.

Suivant votre utilisation de l'EDI vous pouvez en avoir d'autres.

## Foire aux questions

### Q. Quelques raccourcis pour me faire adopter Eclipse ?

R. Si je devais en donner deux :

- CTRL+ / pour ajouter ou enlever des commentaires
- CTRL + i pour la correction de l'indentation

### Q. J'ai mis mes icônes ou mes ressources dans le répertoire bin et elles disparaissent régulièrement ! Que se passe-t-il ?

R. Le répertoire **bin** est un répertoire de travail temporaire. Il faut placer les ressources dans le répertoire **src** et resynchroniser les répertoires (Après cette étape, la synchro est automatique)

### Q : Je veux utiliser une bibliothèque externe livrée sous forme de fichier JAR.

R. Il faut lier la bibliothèque au projet. C'est très facile :

- (M) Project > Properties
- (L) Java Build Path
- (O) Libraries ,
- (B) add Externals Jar

Perso, je garde les bibliothèques spécifiques à un projet dans le répertoire de celui-ci.

### Q. Je voudrais livrer mon application sous forme d'un fichier JAR exécutable. Comment faire ?

R. Pour exporter tout ou partie d'un projet, il suffit de cliquer avec le bouton droit sur celui-ci, de choisir l'action d'exporter. Dans la liste des possibilités, il faut d'abord choisir Java puis Jar File. Sont ensuite affichées des boîtes de dialogues successives pour paramétrer l'export : plusieurs projets ou partie seulement dans une vue hiérarchique avec cases à cocher, la configuration du fichier Manifest qui donne l'emplacement de la méthode main() est sur le dernier écran.



### Q. Je voudrais utiliser un concepteur d'interface graphique, comment puis-je faire ?

R. Le plugin VEP a été archivé pendant les vacances scolaires 2011. Il faut maintenant se tourner vers **WindowBuiler**

Le lien :

- [www.eclipse.org/windowbuilder](http://www.eclipse.org/windowbuilder) pour le plugin

### Q. La compilation en arrière-plan prend beaucoup de ressources pour mon portable sur batterie (ou je voudrais alléger les communications réseaux). Puis-je désactiver cette fonctionnalité ?

R. Oui et non. C'est prévu dans le menu Projet. On peut cocher et décocher **Build automatically**. Eclipse utilise un compilateur interne (et pas javac) pour les erreurs classiques. Ainsi, on a l'impression que désactiver ne marche pas

Sinon l'option (M) Window > Preferences > General > Workspace.

### Q. Je veux générer la documentation au format Javadoc. Comment faire ?

R. L'option se cache dans le menu (M) Project > Generate Javadoc. Pour générer la documentation d'une classe, se méfier du niveau de visibilité des méthodes et attributs : public, protected, package ou private. Pour la génération de la documentation du projet, je ne trouve pas que les boîtes de dialogue soient bien visibles mais il y a des cases à cocher pour les classes et les packages dont il faut générer la documentation