# La plate-forme Eclipse

**Qu'est-ce que c'est ?**

Eclipse est une plate-forme d'aide au développement ***générique*** et ***extensible***, conçue par IBM. En d'autres termes, elle n'est pas dédiée à un type spécifique de développement. Elle permet de mettre en œuvre aussi bien des programmes Java ou C++ que pour accéder à des bases de données relationnelles ou pour effectuer des développements en technologie XML.

La plate-forme est gratuite et portable sur différentes sortes de systèmes (exemples : Windows, Linux, Mac OS..) car entièrement écrite en Java.

Eclipse possède un mécanisme très souple d'extension appelé ***plug-in***

Pour lancer l'environnement Eclipse depuis une station ***Windows***, faire : *Programmes → Langages programmation → Eclipse → Eclipse Java*.

Pour lancer l'environnement Eclipse depuis une station ***Linux***, faire : *Applications → Programmation* et cliquer sur *Eclipse Java*.

**Organisation de l'espace de travail.**

La plate-forme opère sur les fichiers réguliers des utilisateurs. Tous les fichiers sont directement accessibles en dehors d'Eclipse par les outils standards. Ils sont lisibles et localisables par l'utilisateur.

Chaque utilisateur possède son propre ***espace de travail***, divisé en ***projets***.

Chaque projet est un répertoire. Deux répertoires de projet ne peuvent se recouvrir.

Les projets Eclipse peuvent être situés physiquement dans des zones très diverses de la hiérarchie de fichiers de l'utilisateur. L'état de l'espace de travail et de la hiérarchie de projets est maintenu dans un répertoire privé, appelé ***workspace***.

**Visualisation de l'espace de travail (perspectives et vues).**

On appelle plan de travail (***workbench***) la fenêtre principale d'Eclipse. Le plan de travail donne accès à l'espace de travail au moyen d'éditeurs et de ***vues***. Les outils proposés dans le plan de travail seront différents suivant la ***perspective*** visée par l'utilisateur.

En effet, à chaque type de développement est associé une ***perspective***, qui s'obtient par *Fenêtre → Ouvrir la perspective*. Une perspective est un ensemble de fenêtres du plan de travail qui permettent de visualiser et faire évoluer un développement en en modifiant les ressources qui peuvent être des fichiers de code source, des fichiers de code objet, des données, etc.

Pour faire du développement Java, il y a au moins trois perspectives à connaître :

- La perspective ***ressource*** encore appelée perspective ***simple*** : Eclipse se présente à l'image d'un explorer Windows qui permet à l'utilisateur de naviguer et d'intervenir de manière interactive dans une arborescence de fichiers qu'il désire gérer ;
- La perspective ***développement Java*** : Eclipse devient un outil de développement Java qui permet à l'utilisateur d'éditer, compiler, organiser des programmes Java;

- La perspective « ***debugging*** » ***Java*** : Eclipse devient un outil de mise au point interactive d'applications Java.

- **Ouvertures de vues.**

  Chaque perspective possède son aspect visuel propre qui peut être reconfiguré à volonté. En particulier, chaque perspective offre une combinaison prédéfinie de vues et d'éditeurs. Il est possible d'ouvrir une vue qui ne figure pas dans la perspective en cours en sélectionnant *Fenêtre → Afficher la vue* dans la barre de menus principale. L'utilisateur peut créer des ***vues rapides*** pour affecter un raccourci aux vues qu'il utilise fréquemment. Après avoir ajouté une vue dans la perspective en cours, il peut sauvegarder la nouvelle présentation en cliquant sur *Fenêtre → Sauvegarder la perspective sous*.

**Projet Java.**

Un ***projet*** est essentiellement un répertoire de travail. Un projet est typé, c'est à dire qu'Eclipse s'attend selon le type du projet à une certaine organisation du répertoire associé. Par exemple, il existe un type de projet pour Java, pour C++ ou pour XML. Dans ce qui suit, on présente comment créer un projet Java.

Pour Eclipse, un projet Java est associé à un répertoire sur le poste de travail de l'utilisateur. Le répertoire contient une hiérarchie des fichiers associés au projet où on trouve les sources et les objets du projet. Un projet porte un nom qui peut être différent du nom de son répertoire associé. Ce répertoire peut être placé sur la station de travail selon une des deux manières suivantes :

- Le répertoire du projet est un sous-répertoire du répertoire de travail par défaut d'Eclipse dont le nom est ***workspace***. Sélectionner alors la case *Utiliser par défaut* dans la fenêtre de création d'un projet Java (voir ci-dessous)
- C'est un répertoire situé n'importe où ailleurs que dans le ***workspace*** d'Eclipse. C'est la méthode qui doit être préférée. Pour cela, il faut prendre soin de désélectionner la case *Utiliser par défaut* dans la fenêtre de création d'un projet Java et spécifier l'emplacement du répertoire de l'utilisateur sur sa station (voir ci-dessous).

Indépendamment de son emplacement sur la station de travail, le répertoire d'un projet Java et sa hiérarchie peuvent être organisés de deux manières : soit le répertoire racine du projet lui-même est utilisé comme conteneur des sources (projets simples) ; soit le projet comporte des dossiers source à part (projets plus grands).

La section suivante présente les deux méthodes évoquées ci-dessus, mais il est recommandé de toujours utiliser la seconde méthode, qui consiste à séparer les sources dans des répertoires à part car elle permet de structurer plus clairement les projets Java ; de plus, elle est inévitable pour développer des librairies Java.

- **Création d'un projet Java avec ou sans dossier source.**

  Comme indiqué précédemment, pour les projets simples, le projet lui-même est utilisé comme conteneur de source. Pour les projets plus grands, on crée un projet comportant des dossiers source. Les deux méthodes sont présentées conjointement.

  1. Dans la fenêtre principale du plan de travail, cliquer sur *Fichier → Nouveau → Projet*. L'assistant de création d'un projet s'ouvre alors.

2. Sélectionner *Java* dans la sous-fenêtre de gauche et *Projet Java* dans la sous-fenêtre de droite puis cliquer sur *Suivant*. L'assistant de création d'un projet Java s'ouvre alors.
3. Dans la zone *Nom du projet*, saisir le nom à donner au nouveau projet Java.
4. Cocher ou désélectionner la case *Utiliser par défaut* pour indiquer si l'utilisateur souhaite utiliser l'emplacement par défaut pour son nouveau projet (méthode déconseillée). Si cette case est décochée, saisir l'emplacement du projet ou cliquer sur *Parcourir* pour le sélectionner.
5. Cliquer sur *Suivant*. La page *Paramètres Java* s'ouvre alors.
6. Dans l'onglet *Source*, par défaut le projet est le seul dossier source et est également le dossier de sortie par défaut (sauf si l'utilisateur a changé les préférences, manipulation qui n'est pas décrite ici). C'est là qu'intervient la possibilité de désigner un (ou des) sous-répertoire(s) destiné(s) à contenir les sources. Pour ajouter un sous-répertoire source :
   - Cliquer sur le bouton *Ajouter un dossier*. La boîte de dialogue *Sélection du dossier* source s'ouvre.
   - Puis cliquer sur le bouton *Créer un dossier* pour créer un dossier source. L'utilisateur pourra également créer des dossiers source ultérieurement.
   - Sélectionner les dossiers nouvellement créés dans la boîte *Sélection du dossier source* et cliquer sur *OK*.
7. Remplacer éventuellement le nom par défaut figurant dans la zone *Dossier de sortie* par défaut pour que le nom du dossier de sortie soit différent.
8. Cliquer éventuellement sur l'onglet *Projets* et sélectionner les projets à insérer dans le chemin de compilation pour ce projet. N'utiliser ces options que si le projet dépend d'autres projets.
9. Cliquer éventuellement sur l'onglet *Bibliothèques* et sélectionner les fichiers JAR et les dossiers CLASS à ajouter au chemin de compilation pour ce nouveau projet, puis connecter la source aux fichiers JAR. N'utiliser ces options que si le projet nécessite d'autres bibliothèques.
10. Cliquer sur l'onglet *Ordre et exportation* et utiliser les boutons *Haut et Bas* pour déplacer le fichier JAR ou les dossiers CLASS vers le haut ou le bas du chemin de compilation de ce nouveau projet.
11. Après avoir terminé, cliquer sur *Fin*.

**Compilation d'un programme.**

Par défaut, la compilation du code source courant est faite automatiquement au moment de la sauvegarde du fichier. Cette option peut être modifiée en sélectionnant *Plan de travail* dans la sous-fenêtre de l'arborescence pour afficher la page de préférences

correspondante, puis en sélectionnant (ou en désélectionnant) l'option *Effectuer une génération automatique lorsqu'une ressource est modifiée*.

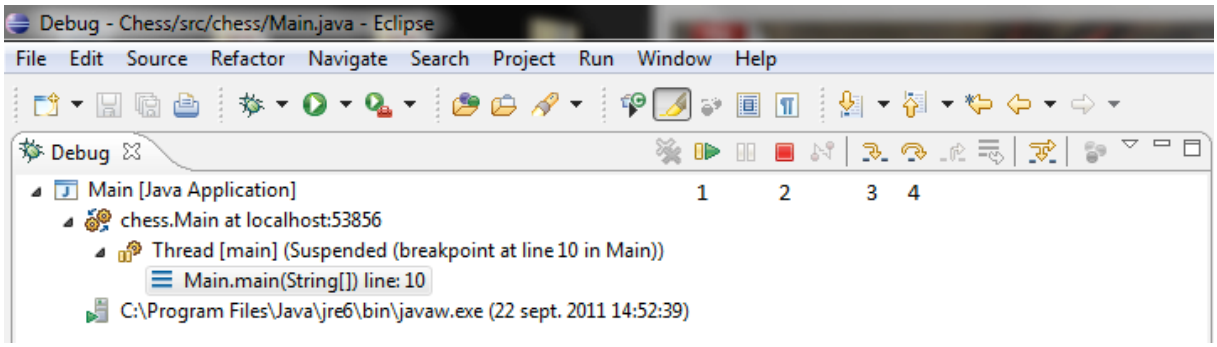
**Exécution et débogage.**

Vous pouvez lancer des programmes Java à partir du plan de travail. Ces programmes peuvent être lancés en mode exécution ou débogage.

- En mode exécution : le programme s'exécute mais l'utilisateur ne peut ni interrompre ni suivre son exécution.
- En mode débogage : l'utilisateur peut suspendre et reprendre l'exécution, inspecter les variables et évaluer les expressions.

Le lancement en mode exécution s'effectue par *Exécuter* → *Exécuter...* Il faut ensuite choisir un nom pour le test et préciser le projet sur lequel porte le test, ainsi que la classe *main* à exécuter. Les arguments habituellement saisis en ligne de commande seront saisis dans la fenêtre *Arguments* → *Arguments du programme***.** Pour finir, *Appliquer* les paramètres et *Exécuter* le test.

Pour déboguer un programme, vous pouvez placer des points d'arrêts (Clic droit dans la barre à gauche de la vue Java → *Toggle Breakpoint*) dans votre code. L'exécution en mode débogage par *Exécuter* → *Debug* ou F11 lancera l'exécution du programme et l'interrompra au premier point d'arrêt rencontré en ouvrant la vue de debug. Dans cette vue, vous pouvez savoir où le programme s'est arrêté, choisir de le relancer jusqu'au prochain point d'arrêt ou jusqu'à la terminaison (1), d'arrêter l'exécution (2) ou choisir de l'exécuter en mode pas à pas en entrant dans les méthodes (3) ou non (4).




*Exemple de la vue Debug dans la perspective Debug*

**Edition de texte, aide à la mise au point du code Java.**

- **Formatage de fichiers ou de portions de code Java.**

  Pour formater du code Java, procéder comme suit :

  1. Utiliser la page des préférences de formatage de code (*Fenêtre* → *Préférences* → *Java* → *Module de formatage de code*) pour indiquer ses préférences pour le formatage de code.
  2. Ouvrir un fichier Java et sélectionner le code à formater. Si rien n'est sélectionné, tout le contenu de l'éditeur sera formaté.
  3. Formater le code en effectuant une des actions suivantes :
     i. sélectionner *Source* → *Format* dans le menu en incrustation de l'éditeur, ou

ii. appuyer sur les touches *Ctrl+Maj+F*, ou

iii. sélectionner *Source* → *Format* dans la barre de menus.

- **Utilisation de l'assistant de contenu /code.**

L'assistant de contenu (également appelé assistant de code) peut être utilisé pour rédiger du code Java ou des commentaires Javadoc.

1. Dans un éditeur, placer le curseur à un emplacement correct sur une ligne de code et effectuer l'une des opérations suivantes :

   i. Appuyer sur *Ctrl+Espace*.

   *ii.* Dans la barre de menus, sélectionner *Edition* → *Assistant de contenu*.

   Si l'éditeur Java trouve des éléments correspondant à cet emplacement, la liste des compléments possibles s'affiche dans une fenêtre flottante. Pour réduire la liste, continuer à taper son code. Pour afficher les informations Javadoc d'un élément de la liste (si elles sont disponibles), placer le pointeur sur cet élément.

2. Utiliser les touches de déplacement du curseur ou le pointeur de la souris pour se déplacer dans ces compléments.

3. Sélectionner un élément de la liste pour compléter le fragment, en effectuant l'une des opérations suivantes :

   i. Sélectionner l'élément et appuyer sur *Entrée*.

   ii. Cliquer deux fois sur cet élément.

   iii. Lorsqu'un élément de la liste est sélectionné, les informations Javadoc le concernant sont disponibles sous forme d'info-bulles. L'utilisateur doit sélectionner un élément de la liste pour que les éventuelles informations Javadoc le concernant soient disponibles.

**Raccourcis** (**CTRL + SHIFT + L** pour afficher la liste des raccourcis dans l'éditeur)

- **Auto-complétion.**

**CTRL + Espace** : Auto-complétion

- **Organisation et formatage du code.**

**CTRL + SHIFT + O** : Organise les imports (ajouts des imports nécessaires et suppression des imports inutiles)

**CTRL + SHIFT + F** : Mise en forme du code (vous pouvez surligner une zone de code pour restreindre le formatage)

**CTRL + SHIFT + I** : Indentation du code (vous pouvez surligner une zone de code pour restreindre l'indentation)

**ALT + SHIFT + R** : Renommer une variable, une méthode ou fonction, une classe, etc.

**CTRL + SHIFT + C** : Commente / décommente les lignes sélectionnées (commentaire encapsulant chaque ligne indépendamment)

**CTRL + SHIFT + /** : Commenter / décommente un bloc de lignes sélectionnées (peut avoir le même comportement que **Ctrl+Shift+C** selon le contexte)

- **Recherche et navigation.**

**CTRL + SHIFT + R** : Recherche d'un fichier dans le workspace

**CTRL + SHIFT + T** : Recherche d'une classe dans le workspace

**CTRL + K** et **CTRL + SHIFT + K** : Positionne le curseur sur l'occurrence suivante ou précédente de la sélection de départ (réagit en fonction du paramétrage courant de la recherche avec **CTRL + F**)

**CRTL + SHIFT + P** : Pour se déplacer d'une accolade à l'autre

- **Affichage d'informations sur le code.**

**CTRL + O** : Affiche les attributs et méthodes de la classe courante

**CTRL + O une deuxième fois** : Ajoute à l'affichage les attributs et méthodes héritées

**CTRL + T** : Affiche l'arborescence d'héritage de la classe courante

**CTRL + 1** : Affiche le 'QuickFix', Eclipse affiche alors une liste d'actions possibles pour corriger l'erreur.