Environnements de développement (intégrés)

JDT (navigation, édition)

Patrick Labatut
labatut@di.ens.fr
http://www.di.ens.fr/~labatut/

Département d'informatique
École normale supérieure

Centre d'enseignement et de recherche en technologies de l'information et systèmes
École des ponts

Transparents inspirés de ceux du cours de Mihaela Sighireanu en 2006-2007.

Plan

JDT[2]

Le JDT est un ensemble d'extensions du plan de travail Eclipse permettant un développement aisé des programmes Java.

Parmi ces extensions, on trouve (liste non-exhaustive) de quoi :

- Créer des paquets[1] et organiser du code,
- Créer des classes, des interfaces, des méthodes,
- Éditer avec indentation automatique, génération de code, avertissements et indication d'erreurs,
- Naviguer dans le code (paquet, classe/interface, méthode, membre),
- Rechercher des définitions, utilisations de classe, méthode,. . . ,
- Exécuter dans un contexte donné,
- Déboguer et exécuter à la volée le code,
- . . .

[1] *package*, en anglais
[2] *Java Development Tools*

Ce qui ne change pas. . .

- Configuration des préférences (claviers, souris, barre d'outils, polices, . . .) à partir de *[Window / Preferences. . . / General]*.
- Création, ouverture, fermeture, suppression de projet (on sélectionne naturellement un projet Java et on importe du code Java à partir du système de fichiers ou d'une archive).
- Rechercher un fichier.
- Sélectionner un espace de travail.
- Comparer des fichiers.
- Marquer des ressources et ajouter des tâches.
- Travailler avec l'historique locale.

Perspectives du JDT

- *Java* : Utilise comme moyen de navigation principale la vue *Package Explorer* qui présente de façon arborescente les éléments du projet (similaire à la vue *Navigator*, mais orientée Java),
- *Java Browsing* : Utilise des vues séparées pour chaque type d'élément du projet,
- *Java Type Hierarchy* : Utilise la vue *Hierarchy* comme vue principale de navigation.
- *Debug* : Cf. cours sur le JDT (debogage).

Dans les tous les cas, il est possible de naviguer dans :

- les projets, les paquets, les unités de compilation (fichiers),
- les classes/interfaces, les membres, les méthodes.

Vues du JDT (1/2)

- *Package Explorer* : Joue le rôle de la vue *Navigator* des projets généraux ; il affiche sous forme arborescente tous les éléments des programmes Java, y compris les bibliothèques JAR utilisées par le projet.
- *Outline Java* : Affiche un résumé du fichier en cours d'édition en termes d'éléments Java. En vue rapide, on peut l'activer par *[Mouse-R / Quick Outline]* sur le fichier ouvert en édition et utiliser le *Menu* pour filtrer, trier, ...
- *Javadoc* : Affiche le commentaire Javadoc correspondant à l'élément selectionné ; la vue apparait également en vue rapide dans l'éditeur.

Vues du JDT (2/2)

- *Hierarchy* : À l'activation (*[Mouse-R / Open Type Hierarchy]* sur un identifiant de classe), affiche la hiérarchie de la classe sélectionnée et la liste les membres. On peut sélectionner la partie de la hiérarchie à voir et les membres affichés en fonction de leurs attributs.
- *Call Hierarchy* : À l'activation (*[Mouse-R / Open Call Hierarchy]* sur un identifiant de méthode), affiche la hiérarchie des appels à la méthode sélectionnée. L'affichage est fait par niveau d'appel, à la demande.

Les autres vues (*Breakpoints*, *Display*, *Debug*, *Expressions*, *Variables*, *JUnit*, ...) seront présentées dans les cours suivants.

Rechercher un élément

Plusieurs solutions :

- En utilisant une vue et en (double-)cliquant sur un élément, on obtient dans l'éditeur sa définition ou son utilisation.
- En utilisant *[F3]* (ou *[Mouse-R / Open]*) sur le nom d'un élément, on obtient sa définition.
- En utilisant *[Mouse-R / References / . . .]* sur le nom d'un élément, on obtient toutes les utilisations de cet élément dans l'espace de travail, le projet, la hiérarchie, . . .
- En utilisant le menu *[Search / Java. . .]* puis en précisant le nom (ou une partie du nom), son type, ... La vue *Search* est ouverte et on peut sélectionner parmi les différents résultats trouvés.

Editeur spécialisé pour le Java

Les outils inclus dans l'éditeur :

- Coloration syntaxique,
- Indentation automatique,
- Assistance à l'écriture de code,
- Assistance à l'importation,
- Détection et assistance à la correction d'erreurs simples,
- Débogueur integré.

Configuration de l'éditeur

Pour sa configuration, utiliser la section appropriée de *[Window / Preferences... / Java / Editor]*.

[Window / Preferences... / Java / Editor / Syntax Coloring] : Permet de gérer les préférences de l'affichage en couleurs de la syntaxe dans l'éditeur.

[Window / Preferences... / Java / Code Style / Formatter] : Permet de gérer l'indentation des programmes Java et de créer des nouveaux styles d'indentation (déconseillé).

[Window / Preferences... / Java / Editor / Folding] : Indique quelles sont les parties du code qui peuvent être « repliées » (⊖ à gauche de l'éditeur).

[Window / Preferences... / Java / Editor / Mark Occurrences] : Indique les endroits du code qui doivent être marqués quand on positionne la souris sur un élément.

Mise en forme du code

[Source / Toggle Comment] : Commente/Décommente la ligne courante.

[Source / Correct Indentation] : Corrige l'indentation de la ligne courante ou de la sélection.

[Source / Format] : Améliore la présentation de la ligne courante ou de la sélection.

[Source / Sort Members. . .] : Trie les membres d'une classe.

[Source / Clean Up. . . / Code Style] : Améliore (en profondeur) la lisibilité du code sans en changer la sémantique.

Aide à l'écriture de code (1/2)

Génération de squelette de code :

- *[File / New... / Class]* : Création d'une classe ; l'assistant permet de préciser les attributs de la classe, de choisir la classe de base et/ou les interfaces à implémenter, de générer des commentaires (vides), des squelettes des méthodes et des constructeurs, ...
- *[Source / Generate Setters and Getters...]* : Création de méthodes de type accesseurs/affecteurs à partir des champs de la classe.
- *[Source / Generate Constructors using Fields...]* : Création de constructeurs à partir des champs de la classe.
- *[Source / Override/Implement Method...]* : Création d'une méthode.
- *[Source / Externalize Strings...]* : Exportation des chaînes de caractères littérales (pour traduire les programmes en différentes langues...).
- *[Edit / Content Assist / Default]* : En fonction du contexte, l'assistant propose des schémas respectant la syntaxe du langage. Par exemple : instructions de contrôle de flot (for, while, if, switch, ...), invocation de méthodes, ...

Aide à l'écriture de code (2/2)

Suggestion/Correction :

- *[Edit / Word Completion]* : Complète le mot courant pour en faire un identifiant ou mot clé valide (permet d'obtenir les complétions envisageables en l'invoquant à nouveau).
- *[Edit / Quick Fix]* : Propose des corrections (pour les erreurs de typo...).
- *[Edit / Content Assist / Default]* : Permet comme *Word Completion* de compléter le mot courant mais de manière plus intelligente (en tenant compte du langage et du contexte), en particulier, indique les méthodes disponibles, la liste des arguments de ces méthodes...
- Compilation du code à la volée (option *[Window / Preferences... / General / Workspace / Build automatically]*, activée par défault) : Eclipse utilise un compilateur interne incrémental qui permet de signaler dynamiquement les morceaux de code incorrects et de fournir des suggestions de correction.

Configuration de l'aide à l'écriture de code

Pour changer le comportement par défaut, aller dans *[Window / Preferences / Java]* puis :

- *[Editor / Code Style / Code Templates]* : Pour génération de code et de commentaires.
- *[Content Assist]* : Pour gérer les insertions/suggestions automatiques.
- *[Editor / Templates]* : Pour la mise en forme du code généré.
- *[Editor / Typing]* : Pour gérer différents comportements automatiques pendant l'édition du code.

Gestion des `import`

Les importations nécessaires à la compilation du code peuvent être générées :

- en cours d'édition,
- après l'edition en utilisant *[Source / Organize Imports. . .]*.

Pour gérer ce comportement, utiliser *[Window / Preferences. . . / Java / Code Style / Organize Imports]*.

Assistance à la réorganisation du code

Cf. Maintenance de projet/Refactorisation.

Tâches

Les tâches sont générées soit :

- explicitement par l'utilisateur,
- implicitement à partir de commentaires prédéfinis (`FIXME`, `TODO`, `XXX`, ...).

La vue *Task* permet de gérer les tâches en lien avec l'éditeur.

Refactorisation

Cf. cours sur la refactorisation.

Génération de documentation

Cf. cours sur la génération de documentation avec Javadoc.

Tests unitaires

Cf. cours sur les tests unitaires avec JUnit et la couverture de code.