Environnements de développement (intégrés)

Introduction aux EDI, la plateforme Eclipse

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

MCours.com

Plan

Définition et composantes

Environnement de développement (intégré)

Un *environnement de développement intégré* (EDI) [1], est un logiciel regroupant un ensemble d'outils nécessaires au développement logiciel dans un (ou plusieurs) langage(s) de programmation.

Outils inclus au minimum dans un EDI :

- un éditeur de texte spécialisé (avec coloration syntaxique, indentation automatique, complétion automatique, . . .),
- un compilateur (ou au moins l'intégration d'un compilateur existant),
- un débogueur (ou au moins l'intégration d'un débogueur existant),
- des outils d'automatisation de la compilation et de gestion de projets.

[1] Integrated Development Environment (IDE) en anglais.

Définition et composantes

Outils souvent également présents :

- un système de gestion de versions (ou l'intégration avec un système existant comme CVS ou Subversion),
- des outils de conception d'interface graphique (IG)[2] (qui génèrent des squelettes de code d'interface graphique à partir d'une description graphique),
- un navigateur de classes (pour explorer la hiérarchie des classes),
- des outils de tests unitaires (vérification systématique du code) et de couverture du code,
- des outils de maintenance/remaniement du code (*refactoring* en anglais),
- un générateur de documentation (ou l'intégration avec un système existant comme Javadoc ou Doxygen).

[2] User Interface (UI), en anglais.

Définition et composantes

But des EDI

Augmenter significativement la productivité du développeur :

- en minimisant le temps passé à basculer entre les différentes tâches intervenant dans le cycle de développement logiciel (édition, compilation, exécution, débogage, test, documentation, import/export vers le dépôt de versions, ...),
- en minimisant le temps d'apprentissage requis par les differents outils intervenant dans le cycle de développement : pas de syntaxe (e.g. celles des `Makefile`), ou de commandes/instructions (e.g. `jdb`/`gdb`) à apprendre.

Évidemment, l'utilisation d'un EDI ne dispense pas complètement d'une certaine familiarité avec chacune des étapes du cycle de développement...

Historique

Rappel sur les progrès réalisés en développement logiciel...

« Préhistoire » :

1950-1960 : cartes perforées,

1960-1970 : apparitions des premiers terminaux et systèmes d'exploitation (SE) :

- éditeurs de texte très basiques (ligne par ligne : `ed` (1969), puis plein écran : `vi`, `EMACS` (1976))
- assembleur,
- compilateur C (1972), et
- débogueur.

1970-1980 : `make` (1977), systèmes d'automatisation de la compilation.

MCours.com

Historique

Avec le développement des SE grand public ayant une interface graphique (1980-1990) et l'augmentation progressive de la taille des programmes, le besoin d'outils de développement plus avancés se fait ressentir et les premiers EDI apparaissent ; quelques dates :

- 1983 : Borland Turbo Pascal, très accessible ($50)
- 1987 : Borland Turbo C
- 1991 : Microsoft Visual Basic
- 1992 : Microsoft Visual C++

DÉMO

Aujourd'hui : développement sans EDI

Le cycle édition/compilation[3]/exécution a lieu dans un éditeur de texte avancé qui (en général) gère les fonctionnalités suivantes (entre autres) :

- coloration syntaxique
- indentation automatique
- complétion automatique
- intégration avec le compilateur (`M-x compile`/`C-x ‘` sous Emacs, mode QuickFix sous Vim, . . .)

Examples :

Emacs : libre, multiplateforme, fonc. : ● ● ●, appr. : ● ○ ○,
Vim : libre, multiplateforme, fonc. : ● ● ●, appr. : ○ ○ ○,
Kate/Kwrite : libre, Unix uniquement, fonc. : ● ● ○, appr. : ● ● ●,
gedit : libre, Unix uniquement, fonc. : ● ○ ○, appr. : ● ● ●,
TextMate : payant, Mac OS X uniquement, fonc. : ● ● ○, appr. : ● ● ○.

[3] avec le compilateur natif : gcc sous Linux. . .

Aujourd'hui : développement sans EDI

Ce type d'approche du développement logiciel est limitée :

- la complétion automatique des éditeurs ne dépend pas (assez) du contexte,
- il n'y a pas ou peu d'intégration avec le débogueur,
- il n'y a pas ou peu d'intégration avec les outils de gestion de version,
- des tâches laborieuses comme la maintenance/remaniement de code restent entièrement manuelles,
- requiert une expertise importante dans chacune des phases du cycle de développement (configuration de l'éditeur de texte, syntaxe des `Makefile`, commandes du débogueur, ...).

Aujourd'hui : développement avec EDI

Logiciels propriétaires :

C++ Builder (Borland) : C/C++, gratuit/payant, Windows uniquement,

JBuilder (Borland) : Java (gratuit/payant, multiplateforme),

Xcode (Apple) : C/C++, Objective C, Java (payant, Mac OS X seulement),

Visual Studio (Microsoft) : C/C++, C#, Web (gratuit/payant, Windows uniquement),

... : -

Aujourd'hui : développement avec EDI

Logiciels libres :

KDevelop (KDE) : C/C++, Java, basé sur les outils GNU (GCC, make, GDB) et sur d'autres outils répandus (CVS, Doxygen) (Unix uniquement),

Anjuta (GNOME) : C/C++ uniquement, aussi basé sur les outils GNU (multiplateforme),

Netbeans (Sun) : initialement Java uniquement, maintenant C/C++, Java Micro Edition[4], Ruby, JavaScript, ...(multiplateforme),

Eclipse : Java, C/C++, Java Micro Edition, PHP, ...(multiplateforme).

Pourquoi choisir Eclipse ? Java, libre, multiplateforme.
DÉMO

[4] Java pour terminaux mobiles.

Un EDI (de plus) pour Java ?

Initialement un simple EDI Java (IBM/OTI VisualAge for Java), Eclipse est devenu un EDI pour développer des EDI et d'autres outils.

Objectif

Offrir une plateforme ouverte pour le développement d'applications :

- non-dédiée à un langage ou un SE ou une IG,
- facile à comprendre mais aussi facile à étendre,
- paramétrable selon les besoins/goûts du développeur,
- capable d'automatiser les tâches lourdes du développement,
- ayant une base stable,
- utilisable pour son propre développement[5],
- promouvoir l'utilisation de Java.

[5] bootstrap-able, en anglais.

Origine et ressources

1996 :	IBM achète OTI qui développe la suite d'EDI VisualAge (en SmallTalk), et en particulier VisualAge for Java,
Nov. 2001 :	lancement du projet libre Eclipse et création de l'Eclipse Consortium,
Jan. 2004 :	création de l'Eclipse Foundation, organisation à but non lucratif qui regroupe plusieurs grandes entreprises (AMD, Borland, IBM, Intel, Motorola, Nokia, Oracle, . . .) dans le but de conduire le dévelopement de la plateforme Eclipse.
. . . :	-
Juin 2006 :	Eclipse 3.2 (depuis, une nouvelle version par an)
Juin 2007 :	Eclipse 3.3
Juin 2008 :	Eclipse 3.4

Plateforme Eclipse

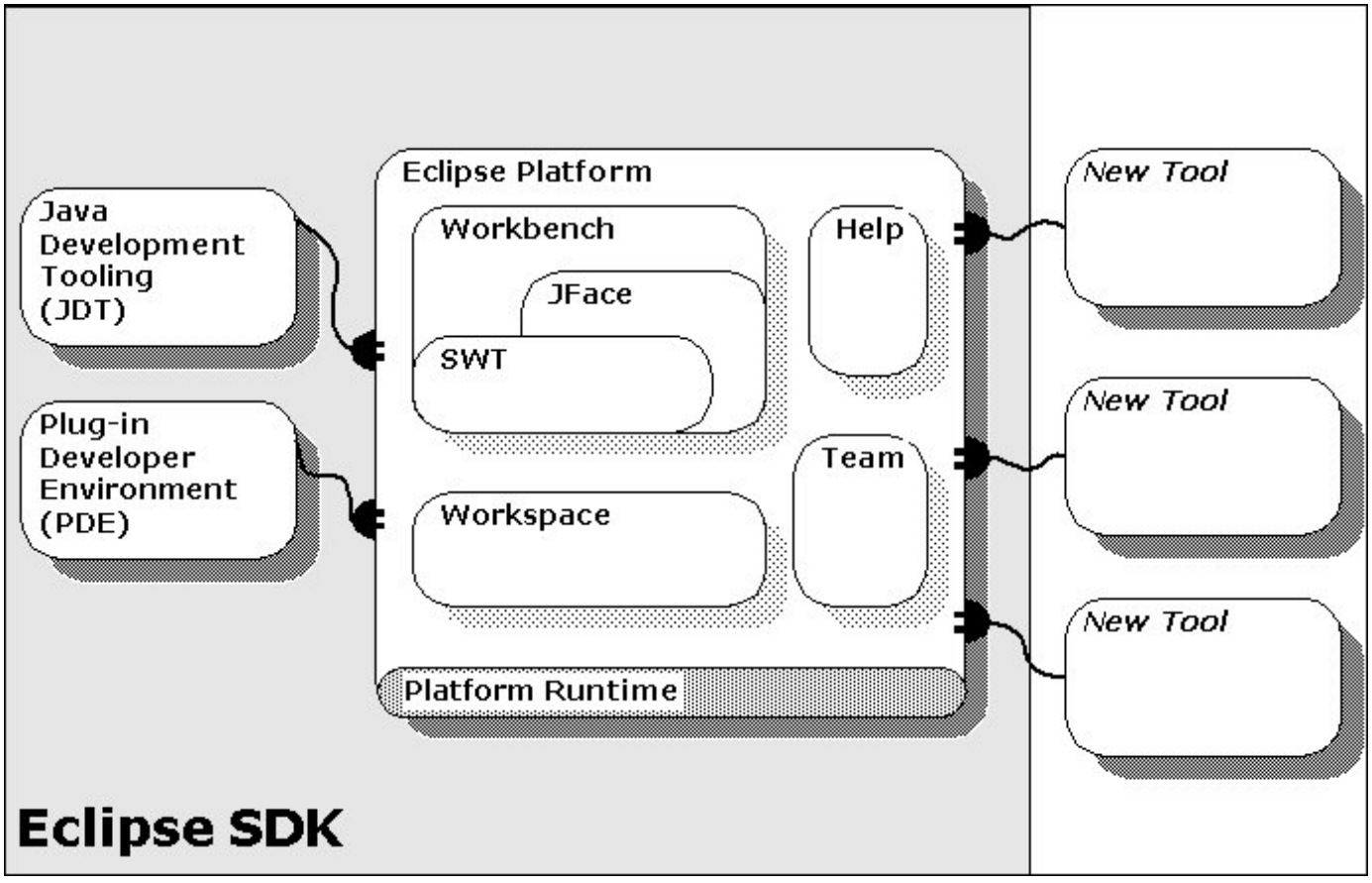
Eclipse = plateforme + greffons

- plateforme :
  - un exécutif[6] indépendant du SE (JVM),
  - un ensemble basique de greffons[7] extensibles,
  - des mécanismes (API), règles et outils pour construire des greffons,
  - un moteur pour découvrir, charger et exécuter des greffons.
- greffon : la plus petite unité qui peut être développée et utilisée séparément ;
  - se connecte à des points précis de la plateforme,
  - remplit une tâche (pas forcement exécutable),
  - offre des points d'extension,
  - coexiste avec d'autres greffons,
  - instance : ensemble de greffons qui coopèrent pour offrir un EDI.

MCours.com

[6] runtime, en anglais
[7] plug-in, en anglais

Architecture de la plateforme Eclipse



Exécutif

- Définit les points d'extension et le modèle de greffon.
  - point d'extension $\longrightarrow$ interface
  - greffon = archive JAR + interfaces implémentées + interfaces utilisées
  - déclaration de greffon = manifeste (dépendences à l'exécution) + interface (type)

  DÉMO
- Découvre dynamiquement les greffons et maintient un registre d'informations sur les greffons et sur leurs points d'extension.
- Charge à la demande les greffons.
- Met à jour automatiquement les instances.

Gestion des ressources : espace de travail[8]

- Ressources : fichiers, répertoires, projets, etc. . . ,
- Espace de travail = un ou plusieurs projets,
- Projet = partie du système de fichiers qui a une “personnalité” (définie par les plug-ins).
  Exemples : projet Java, projet de greffon, projet Web,
- Implémente un mécanisme d’historique local pour suivre les changements des ressources. DÉMO

[8] workspace, en anglais.

Plan de travail[9]

- Fournit l'IG pour l'utilisateur de la plateforme.
- Spécificité Eclipse : l'IG a l'apparence d'une application native du SE et est basée sur deux outils (SWT – Standard Widget Tool, JFace) qui peuvent aussi être utilisés directement.
- Composantes physiques de l'IG : menus, barre d'outils, boutons, onglets, fenêtres.
- Composantes logiques de l'IG (paramétrable par des greffons) :
  - Éditeur : ouvre, modifie et sauvegarde des objets ; lance des actions.
  - Vue : fournit des informations sur les objets (structure, composantes, etc. . .) en communiquant avec d'autres vues ou éditeurs.
  - Perspective : ensemble d'éditeurs et vues ayant une disposition précise dans le plan de travail.
    Exemples : navigation, documentation, débogage, etc. . .

  DÉMO
- C'est le point d'extension le plus étendu !

[9] workbench, en anglais

Développement collaboratif

- Contrôle les versions et le partage d'un projet entre différents développeurs.
  - enregistre dans une archive,
  - gère des modifications de fichiers,
  - récupère toute modification enregistrée,
  - permet de visualiser les différences entre les versions,
- Le système CVS (Concurrent Version System) est utilisé par défaut.
- API pour l'interface avec d'autres systèmes (Subversion par exemple).

Serveur d'aide

- Définit des points d'extensions pour la documentation en ligne.
- Sert de base pour le système d'aide d'Eclipse.

Installer/Utiliser Eclipse

- Installer un JDK (de Sun...) récent (Windows/Linux : `http://java.sun.com/javase/downloads/`, Mac OS X : cf. le site web d'Apple).
- Télécharger l'archive Eclipse Classic ("Eclipse Platform, Java Development Tools, and Plug-in Development Environment") qui correspond à votre SE sur `http://www.eclipse.org/downloads/`[10].
- Extraire l'archive à l'endroit habituel pour votre SE.
- Lancer l'exécutable extrait de l'archive (`eclipse` ou `eclipse.exe`).

[10] La dernière version est la 3.3 mais celle installée dans les salles informatiques est la 3.2. Aussi, il serait peut-être préférable d'installer la 3.2 pour des raisons de compatibilité des projets. Elle est disponible sur : `http://archive.eclipse.org/eclipse/downloads/drops/R-3.2.2-200702121330/`

Installer/Utiliser Eclipse

- (Éventuellement) régler les paramètres (quelques exemples) :
  - la machine virtuelle utilisée :
    `eclipse -vm machine_virtuelle`
  - les paramètres de la machine virtuelle (ici la quantité maximum de mémoire à utiliser, par défaut 256 Mo) :
    `eclipse -vmargs -Xmx512`
  - l'espace de travail utilisé (par défault, `/workspace` sous Unix)
    `eclipse -data espace_de_travail`
- (Éventuellement) Vérifier le réglage :
  Help / About Eclipse SDK / Configuration Details

Plan du cours

- Développement en Java avec Eclipse JDT :
  - développement classique (éditer/compiler/exécuter)
  - déboguer
  - tests unitaires en Java (avec JUnit)
  - développement collaboratif (avec CVS)
  - compilation (avec Ant)
  - création de documentation (Javadoc)
- Développement de greffons avec Eclipse PDE.

Pratique

- Informations/Supports de cours sur :
  `http://www.di.ens.fr/~labatut/`
  Enseignements/Environnements de développement
- Cours et TP (normalement) en alternance une semaine sur deux (consulter régulièrement la page web. . .).
- TP : exercices de programmation Java avec Eclipse.
- Examen : sur machine, exercices similaires à ceux des TP.
- Projet : application plus complexe à développer/étendre sous Eclipse avec les outils vues en cours et TP (avec une soutenance).
- Note finale : $\frac{\mathrm{N_{examen}}+\mathrm{N_{projet}}}{2}$

MCours.com