Environnements de développement

Mihaela Sighireanu

UFR d'Informatique Paris 7, LIAFA, 175 rue Chevaleret, Bureau 6A7
http://www.liafa.jussieu.fr/∼sighirea/cours/edi/

Cours 1

Résumé

Définition et composantes

Definition (EDI)

Un environnement de développement intégré, EDI (ou IDE en anglais), est un logiciel regroupant un ensemble d'outils nécessaires au développement des applications dans un langage $\mathcal{L}$ de programmation.

Exemples d'outils inclus dans un $\mathcal{L}$-EDI :

- un éditeur de texte spécialisé,
- un compilateur,
- un débogueur,
- des outils automatique de gestion d'applications ayant plusieurs fichiers source (projets),
- un gestionnaire de versions et des sauvegardes,
- un générateur de documentation.

Historique

Préhistoire :

- 1950-60 : cartes perforées
- 1960-70 : terminaux, éditeurs de texte basique, compiler et déboguer en ligne de commande.
- 1970-80 : introduction des makefiles et des fichiers de configurations permettant de contrôler convenablement la compilation.

Avec le développement des SE ayant une interfaces graphiques (1980-90), les premiers EDI apparaissent (1981 Turbo Pascal).

Quelques dates :

- 1983 : Borland Turbo Pascal (DOS), prix démocratique (50$)
- 1987 : Borland Turbo C
- 1991 : Microsoft Visual Basic 1
- 1997 : Microsoft Visual Studio (C++)

Exemples

Logiciels libres :

- Emacs, XEmacs : basique, mais adaptables à tout langage
- OpenOffice.org : langages de script
- Kdevelop (KDE) : C, C++, basé sur les outils GNU
- Netbeans (Sun) : initialement conçu pour Java, maintenant C, C++, XML et HTML.
- Eclipse (OTI-IBM) : Java, C/C++, PHP, HTML, etc.

Logiciels propriétaires :

- Visual Studio (Microsoft) : C/C++, .NET, C#, etc.
- JBuilder (Borlans) : Java
- JCreator : Java
- WinDev (PC Soft) : application PC Pocket et Mobile

Un EDI (de plus) pour Java ?

Conçu sur la base d'un EDI Java (VA4J), Eclipse devient un EDI pour développer des EDIs et d'autres outils.

Objectif : offrir une plateforme ouverte pour le développement d'applications.

- non-dédiée à un langage ou SE ou UI
- facile à comprendre mais aussi facile à étendre
- paramétrable selon les besoins/goûts du programmeur
- capable d'automatiser les taches lourdes du développement
- ayant une base stable
- utilisable pour son propre développement (*bootstrap*-able)
- promouvoir l'utilisation de Java

Sources et ressources

1996 : IBM achète OTI qui développe la suite d'EDI Visual Age (en SmallTalk), et en particulier VA4J.

2001 : après un investissement de 40 M$, IBM lance Eclipse 1, grand succès populaire car ouverte et gratuite (licence CPL). Le consortium Eclipse est crée (IBM, Borland, RedHat, SuSE, Intel,...)

2007 : Eclipse 3.2

Bibliographie :

- `www.eclipse.org` (cours, API, etc.)
- Le manuel (très complet, HTML) inclut dans la distribution.
- Steve Holzner, Eclipse. O'Reilly 2004

Plateforme Eclipse

Eclipse = plateforme + *plug-ins*

- plateforme
  - un exécutif (*run-time*) indépendant du SE (JVM)
  - un ensemble basique de *plug-ins* extensibles
  - mécanismes (API), règles et outils pour construire de *plug-in*
  - un moteur pour découvrir, charger et exécuter des *plug-ins*
- *plug-in* = la plus petite unité qui peut être développé et utilisée séparément
  - se connecte à un point précis de la plateforme
  - remplit une tache (pas forcement exécutable)
  - offre des points d'extension
  - coexiste avec d'autres *plug-ins*
  - instance (*feature*) = ensemble de *plug-ins* qui coopèrent pour offrir un EDI

Plateforme Eclipse

Eclipse = plateforme + *plug-ins*

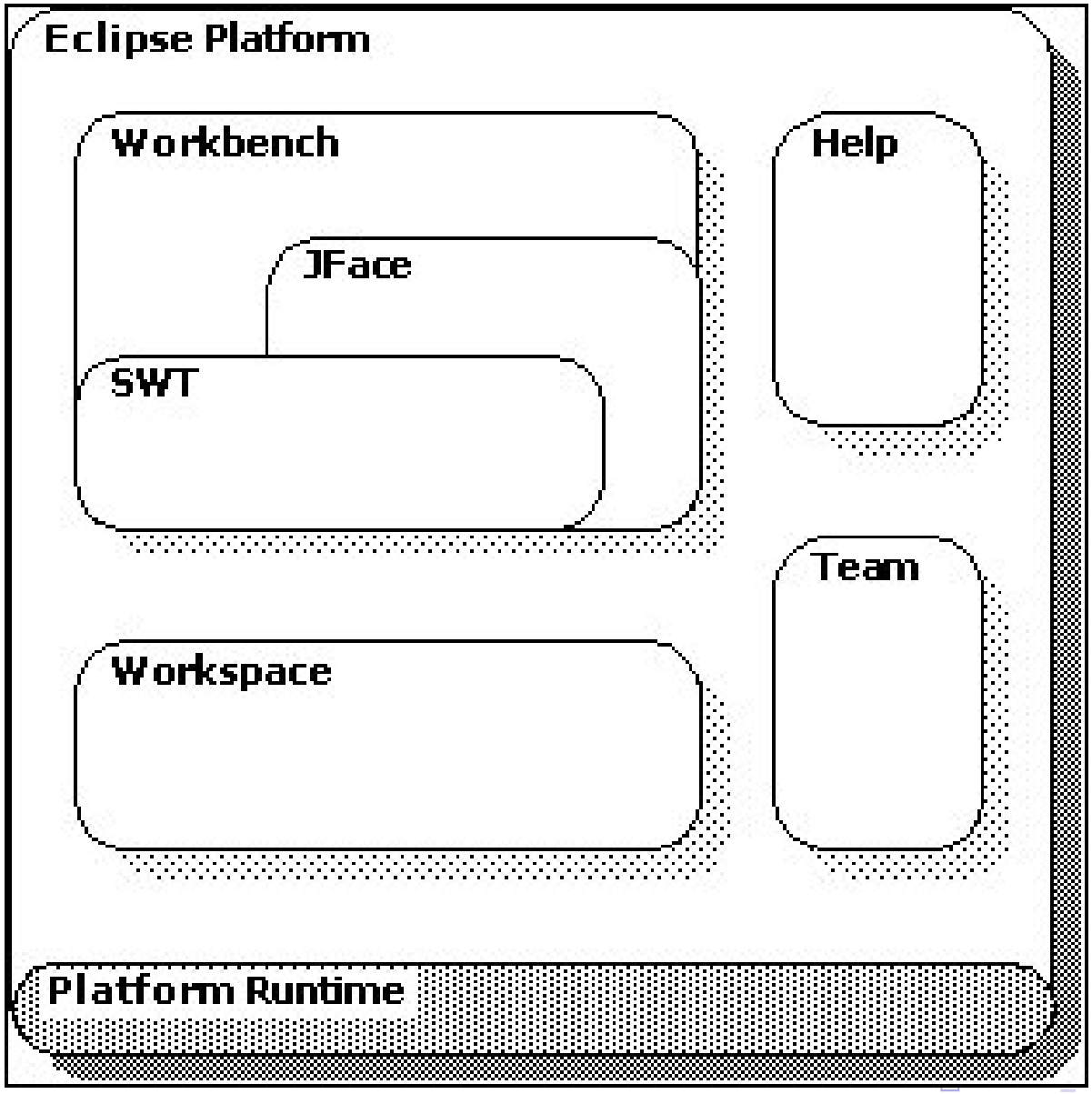
- plateforme
  - un exécutif (*run-time*) indépendant du SE (JVM)
  - un ensemble basique de *plug-in*s extensibles
  - mécanismes (API), règles et outils pour construire de *plug-in*
  - un moteur pour découvrir, charger et exécuter des *plug-in*s
- *plug-in* = la plus petite unité qui peut être développé et utilisée séparément
  - se connecte à un point précis de la plateforme
  - remplit une tache (pas forcement exécutable)
  - offre des points d'extension
  - coexiste avec d'autres *plug-in*s
  - instance (*feature*) = ensemble de *plug-in*s qui coopèrent pour offrir un EDI

Plateforme Eclipse

Eclipse = plateforme + *plug-ins*

- plateforme
  - un exécutif (*run-time*) indépendant du SE (JVM)
  - un ensemble basique de *plug-in*s extensibles
  - mécanismes (API), règles et outils pour construire de *plug-in*
  - un moteur pour découvrir, charger et exécuter des *plug-in*s
- *plug-in* = la plus petite unité qui peut être développé et utilisée séparément
  - se connecte à un point précis de la plateforme
  - remplit une tache (pas forcement exécutable)
  - offre des points d'extension
  - coexiste avec d'autres *plug-in*s
  - instance (*feature*) = ensemble de *plug-in*s qui coopèrent pour offrir un EDI

Architecture de la plateforme Eclipse



Exécutif

(*Platform Runtime*)

- Exécute la JVM.
- Définit les points d'extension et le modèle plug-in.
  - point d'extension = interface
  - plug-in = interfaces implémentées + archive Jar + interfaces utilisées
  - déclaration de plug-in = manifeste (dépendences à l'exécution) + interface (type)

  Démo...
- Découvre dynamiquement les plug-ins et maintient une base sur la base de leur déclaration.
- Charge les plug-ins à la demande.
- Mise à jour automatique des instances (*features*).

Management des ressources : espace de travail

(*Workspace*)

- Ressources : fichiers, répertoires, projets, etc.
- Espace de travail = un ou plusieurs projets.
- Projet = partie du système de fichiers (FS) qui a une personnalité (définie par les plug-ins). Exemples : projet Java, site Web.
- Implémente un mécanisme d'histoire locale (*backup*) pour tracer les changements des ressources. Démo...

Plan de travail

(*Workbench*)

- Fournit l'interface visuelle pour l'utilisateur de la plateforme (UI).
- Spécificité Eclipse : l'UI a l'apparence d'une application native du SE et est basée sur deux outils (SWT – Standard Widget Tool, JFace) qui peuvent être utilisés directement.
- Composantes physiques de l'UI : menus, barre d'actions, boutons, onglets, fenêtres.
- Composantes logiques de l'UI (paramétrable par des plug-ins) :
  - Éditeur : ouvre, modifie et sauvegarde des objets ; active 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, debug, etc.
- Le plus étendu point d'extension !

Plan de travail

(*Workbench*)

- Fournit l'interface visuelle pour l'utilisateur de la plateforme (UI).
- Spécificité Eclipse : l'UI a l'apparence d'une application native du SE et est basée sur deux outils (SWT – Standard Widget Tool, JFace) qui peuvent être utilisés directement.
- Composantes physiques de l'UI : menus, barre d'actions, boutons, onglets, fenêtres.
- Composantes logiques de l'UI (paramétrable par des plug-ins) :
  - Éditeur : ouvre, modifie et sauvegarde des objets ; active 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, debug, etc.
- Le plus étendu point d'extension !

Plan de travail

(*Workbench*)

- Fournit l'interface visuelle pour l'utilisateur de la plateforme (UI).
- Spécificité Eclipse : l'UI a l'apparence d'une application native du SE et est basée sur deux outils (SWT – Standard Widget Tool, JFace) qui peuvent être utilisés directement.
- Composantes physiques de l'UI : menus, barre d'actions, boutons, onglets, fenêtres.
- Composantes logiques de l'UI (paramétrable par des plug-ins) :
  - Éditeur : ouvre, modifie et sauvegarde des objets ; active 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, debug, etc.
- Le plus étendu point d'extension !

Support d'équipe

(*Team support*)

- 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
  - visualise les différences entre les versions
- CVS (Concurrent Version System) est utilisé par défaut.
- API pour l'interface avec d'autres systèmes.

Serveur d'aide

(*Help system*)

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

Utiliser Eclipse

- Installer Java (GNU ou Sun).
- Telecharger l'archive (plateforme + extensions) qui correspond à votre SE sur `www.eclipse.org`; desarchiver.
- Lancer l'exécutable extrait de l'archive (`eclipse` ou `eclipse.exe`).
- Paramétrer (quelques exemples) :
  - la machine virtuelle utilisée :
    `eclipse -vm vmPath`
  - le paramètres de la machine virtuelle, ici la mémoire à utiliser (par défaut 256Mo) :
    `eclipse -vmargs -Xmx512`
  - l'espace de travail utilisé
    `eclipse -data wsPath`
- Visualiser le paramétrage : Help → About Eclipse SDK → ... Details

Plan du cours

- Eclipse et Java (JDK) :
  - développement classique (éditer, compiler, exécuter)
  - déboguer Java (gdb pour déboguer C/C++)
  - test unitaire en Java avec JUnit
  - travail en équipe (CVS)
  - compilation avec Ant
  - création de documentations en ligne (Javadoc)
- Développement de plug-ins avec PDE Eclipse.

Pratique

- En TP : exercices de programmation Java.
- Projet : application plus complexe à développer sous Eclipse.
- Examen : sur les machines

Note finale : (Exam + Projet) / 2