Outils de développement Java sous Linux

Michel CASABIANCA - casa@sweetohm.net

Linux est réputé pour ses outils de développement, mais qu'en est-il au juste des outils Java ? Dans cet article, l'auteur se propose de faire le tour des outils de développement Java disponibles sous Linux et de répondre à cette question: "Linux terre promise pour le développeur Java ?".

Table des matières

Les versions mises à jour de ce document sont disponibles http://www.cafebabe.net.Les formats suivants sont disponibles :

HTML	http://www.cafebabe.net/html/java-linux.html
PDF	http://www.cafebabe.net/arc/java-linux.pdf

Pourquoi Java sous Linux ?

Cet article se veut un panorama des outils de développement Java sous Linux. Le nombre d'outils étant important, je me suis cantonné à ceux qui me sont apparus incontournables, qu'ils soient libres ou non. S'il vous semble que j'ai omis un outil indispensable, merci de m'envoyer un email pour que je corrige mon oubli.

Cette présentation intéressera tout particulièrement les développeurs Java travaillant sous d'autres environnements (comme Windows) et qui envisagent d'en changer. Linux est alors une plateforme de développement intéressante pour sa stabilité, la richesse de son environnement de développement (éditeurs, outils de gestion des versions, make, etc) et son ouverture. Cette migration est d'autant plus tentante que Java est multiplateforme, donc le programmeur ne perd pas le bénéfice de ses développements passés.

Des développeurs travaillant déjà sous Linux peuvent aussi se laisser tenter par ce langage plein de qualités. L'adaptation à Java est facilité pour les habitués du langage C (dont Java s'inspire fortement en ce qui concerne la syntaxe) ou mieux C++ (orienté objet comme Java). Cependant, la migration n'est pas aussi simple qu'elle pourrait sembler au premier abord : si la syntaxe est très proche, les habitudes de programmation sont assez différentes et l'étendue de l'API demande un temps d'adaptation important.

Machines virtuelles

Une machine virtuelle est nécessaire pour exécuter un programme Java (applet ou application). Les

navigateurs compatibles Java incluent leur propre machine virtuelle et il n'est donc pas nécessaire d'en installer une pour visualiser des applets.

Cette partie traite essentiellement des portages pour Linux sur plateforme Intel (désolé pour les autres, mais ne disposant que d'une telle architecture, je ne peux parler d'expérience des autres portages). J'ai néanmoins inclus un tableau récapitulatif où sont référencées les VM disponibles pour Linux sur Alpha, PPC, Sparc et ARM.

JDK de Sun

Tous les portages Linux présentés ci-dessous sont issus de la version Solaris du JDK de Sun. Cette version présente certaines particularités par rapport à la version Windows, que nous allons maintenant détailler :

Particularités du portage Linux

Les portages du JDK sous Linux étant issus de la version Solaris, ils en sont très proches et l'on consultera la documentation des outils pour Solaris (et non Windows). D'autre part, par rapport à la version Windows, une VM Unix présente les particularités suivantes :

Lorsqu'on télécharge la machine virtuelle, les sites FTP proposent parfois deux versions (répertoires), suivant la bibliothèque C installée sur sa machine:

1. *libc5:* pour les anciennes distributions (antérieures à avril 98 environ).
2. *glibc:* si vous disposez d'une distribution récente. Dans ce cas, il est aussi possible d'utiliser la version basée sur libc5.

Le JDK existe aussi parfois en deux versions relatives à la gestion des threads:

1. *Native:* À chaque thread correspond un processus du système.
2. *Green:* La machine virtuelle tourne dans un processus, quelque soit le nombre de threads.

Le choix du type de threads implantés dans la VM est particulièrement important pour les machines SMP (Symetric Multi Processing). Une VM implantant des green threads n'utilisera qu'un seul processeur (elle tourne dans un seul processus, quelque soit le nombre de threads tournant dans la VM) alors que la version utilisant des threads natifs peut utiliser plusieurs processeurs (chaque thread Java engendre un processus différent, ou plus exactement, des pthreads c'est à dire des processus partageant le même espace mémoire). On notera cependant qu'il existe une limite sur le nombre threads natifs pouvant tourner dans une VM. Elle est liée à la limite sur le nombre de processus par utilisateur du noyau Linux, mais il est possible de recompiler son noyau pour l'augmenter [1].

Le JDK comporte trois variations de la machine virtuelle, suivant le mode de liaison avec Motif :

1. Une version *sans AWT*. Elle permet de lancer plus rapidement des applications non graphiques. Pour lancer la VM avec cette option, il faut affecter à la variable d'environnement `NS_JAVA` une chaîne de longueur non nulle (par exemple `export NS_JAVA="true"`).
2. Une version *liée dynamiquement* à Motif. Pour la lancer, affecter à la variable d'environnement `DYN_JAVA` une chaîne de longueur non nulle (par exemple `export DYN_JAVA="true"`).
3. Une dernière version *liée statiquement* aux bibliothèques Motif. Pour la lancer, il suffit qu'aucune des deux variables d'environnement `NS_JAVA` et `DYN_JAVA` ne soit définie (on pourra s'en assurer avec `unset NS_JAVA` et `unset DYN_JAVA`).

A chaque mode de liaison à Motif correspond un exécutable dans un sous répertoire du répertoire `bin` de votre répertoire d'installation du JDK ou du JRE. Ils sont appelés par les scripts du répertoire `bin` qui testent les variables d'environnement avant de choisir l'exécutable à lancer.

Ces machines virtuelles demandent toutes (à l'exception des versions 1.0 et 1.1 de Blackdown) une distribution récente. En effet, elles requièrent :

1. Un noyau 2.2
2. Une glibc en version 2.1

Nous allons maintenant voir dans le détail l'offre en matière de machines virtuelles. Pour Linux, on compte pas moins de trois distributeurs de machines virtuelles :

Blackdown

- http://www.blackdown.org/java-linux/ports.html
- http://khendricks.ivey.uwo.ca/linuxppc/jdk
- http://java.sun.com/products/jdk/1.2/LICENSE

Blackdown est une organisation visant à porter le JDK sous Linux, et a été longtemps la seule à s'intéresser à cette plateforme (avant la prise en compte du marché Linux par Sun et IBM).

Blackdown propose sur son site des versions du JDK de la version 1.0.2 à la 1.2.2 (en version RC4 pour le moment). Ces VMs sont de qualité très inégale qui croit généralement avec le numéro de version.

JDK 1.0.2

La version 1.0.2 (dont le portage a été réalisé par Randy Chapman) a le mérite d'exister, mais est très peu fiable. À éviter donc, d'autant plus qu'il y a peu d'intérêt à utiliser encore un JDK 1.0 (les navigateurs eux-mêmes sont tous munis de machines virtuelles 1.1).

JDK 1.1

Les versions 1.1 (de la 1.1.6 à la 1.1.8) sont de bonne qualité, mais sont livrées sans JIT. On pourra leur adjoindre un JIT Open Source comme TYA ou ShuJIT, mais leur vitesse reste inférieure d'un facteur quatre environ à de bonnes VM 1.1, comme celle d'IBM par exemple (dont le JIT est excellent). Un JRE (Java Runtime Environment) est aussi disponible au téléchargement [2].

JDK 1.2.2

La version 1.2.2 est de très bonne qualité (actuellement en version RC 4) et livrée avec un JIT de Sun (libsunwjit) de bonne facture. Il est possible de la faire tourner avec des green threads ou des threads natifs. Par défaut, elle utilise des threads natifs, mais on peut utiliser des green threads en passant l'argument `-green` à la VM. Blackdown propose aussi un JRE, un Java PlugIn [3] et une version de débuggage pour débugger des librairies JNI ou envoyer un rapport de bug à Blackdown. Pour finir, son JDK intègre le JPDA [4], alors qu'il faut le télécharger séparément pour les versions Windows et Solaris.

Blackdown a aussi réalisé le portage de Java3D pour Linux en version 1.1.3. Pour l'utiliser, il faut avoir installé une implantation de OpenGL (comme Mesa 3.1 à l'adresse http://www.mesa3d.org).

Sun

- http://java.sun.com/products/jdk/1.2
- http://java.sun.com/products/jdk/1.2/LICENSE

Sun propose maintenant un portage Linux de son JDK 1.2.2. Cette version est de très bonne qualité (maintenant en version *Production release*). Il est très proche de la version de Blackdown (dont il est issu), les différences notables étant :

- Il ne comporte pas de JIT. Sun recommande d'utiliser le http://www.inprise.com/jbuilder/linux/download/ (et pas le leur intégré dans le JDK Blackdown ?).
- Il utilise les green threads (les threads natifs ne sont pas supportés dans cette version).

IBM

- http://www.ibm.com/java/jdk/118/linux
- http://alphaworks.ibm.com/tech/linuxjdk
- International License Agreement for Evaluation of Programs

IBM a été (en son temps) pionnier pour le développement de machines virtuelles pour Linux. La sortie de sa version 1.1.6 a fait sensation dans la communauté des développeurs Java sous Linux.

IBM propose deux machines virtuelles pour Linux :

- Une 1.1.8 de bonne facture (en version finale).
- Une 1.3 encore en version bêta (mais cette version du JDK est encore en bêta pour les versions Windows et Solaris).

Ces VMs sont remarquables par la qualité de leur JIT. Mes tests me font penser qu'il est plus lent au lancement, mais que les optimisations de code sont plus poussées ce qui donne de meilleures performances pour des applications ayant une durée de vie dépassant les quelques secondes [5].

Les VM d'IBM utilisent les threads natifs. D'après l'expérience de certain utilisateurs, les applications comportant de nombreux threads auraient cependant tendance à bloquer X en s'accaparant toutes les ressources CPU. Il semblerait que cela soit dû à la priorité accordée aux processus associés aux threads.

IBM met aussi à disposition une implantation de l'API JavaComm permettant d'accéder aux ports série et parallèle de l'ordinateur.

Linux sur Alpha, PPC, Sparc et ARM

Voici un tableau récapitulatif des ports pour Linux sur ces différentes architectures :

Ces ports ont été réalisés pour l'essentiel par http://www.blackdown.org/java-linux/ports.html (sauf la version 1.1.8 et 1.2 pour PPC qui a été portée par http://khendricks.ivey.uwo.ca/linuxppc/jdk/ qui s'est associé avec Blackdown pour ses futurs travaux et la version 1.1.6 pour Sparc qui a été réalisée par Johan Vos). La VM pour Linux sur Alpha existe en version 1.1.7 (et il faut un peu chercher sur le site de Blackdown pour en trouver la trace).

La liste des portages du JDK (toutes plateformes confondues) se trouve à l'url http://java.sun.com/cgi-bin/java-ports.cgi. On peut trouver une documentation sur l'installation d'une VM sous Linux (et bien

d'autres documentations sur Linux) sur le site http://www.linux-france.org/article/devl/java-linux.html.

Kaffe

- http://www.kaffe.org
- Licence http://www.gnu.org/copyleft/gpl.html

Kaffe est une implémentation libre de machine virtuelle Java. Elle est livrée notamment avec la distribution RedHat (dans ses versions 5.x et 6.x). Elle a été portée sur de *très* nombreuses plateformes (pas moins de 43, dont 33 avec JIT !). La liste des plateformes supportées est disponible à l'adresse http://www.transvirtual.com/products/platforms.html.

Les performances ne sont malheureusement pas au rendez-vous malgré la présence d'un JIT. Elles se situeraient au niveau de la machine virtuelle de Sun *sans* JIT d'après les tests des auteurs. Malgré des progrès certains ces derniers mois, la fiabilité n'est pas encore son point fort et de nombreuses fonctionnalités n'ont pas encore été implémentées (voir la liste à http://www.kaffe.org/develop.html). Manquent notamment à l'appel le modèle de sécurité (aucune forme de sécurité n'est présente dans la machine virtuelle) et RMI.

D'autre part, bien que sous licence GPL, Transvirtual a implanté les extensions propriétaires de Microsoft en collaboration avec ce dernier. Ces "fonctionnalités" laissent perplexe et ne sont pas pour améliorer l'image de Transvirtual auprès des développeurs Java pour lesquels la portabilité est une qualité essentielle du langage. Pour plus de détails sur cette affaire, on pourra consulter l'article http://www.zdnet.com/zdnn/stories/news/0,4586,2275724,00.html.

On comprendra que cette machine virtuelle (bien qu'élue meilleure machine virtuelle de l'année 1998 par http://www.javaworld.com) n'est pas encore à la hauteur malgré son exceptionnelle portabilité. Si le numéro de version 1.0 semble usurpé, on peut cependant espérer des améliorations sensibles qui en feront une alternative intéressante à l'offre des autres fournisseurs.

Japhar

- http://www.japhar.org
- Licence http://www.gnu.org/copyleft/lgpl.html

Une autre machine virtuelle sous licence LGPL disponible sur de nombreuses plateformes (voir http://www.japhar.org/supported.html). Malgré des http://sourceware.cygnus.com/mauve favorables, la compatibilité Java semble très approximative. Sa jeunesse (la version testée est numérotée 0.08) peut expliquer sa faible fiabilité. De plus, il manque de toute évidence des paramètres aussi indispensables que la taille maximale de la mémoire allouée (option `-mx` sur la ligne de commande).

Nécessite les classes du JDK de Sun (fichier `classes.zip` du répertoire `/lib`). Japhar pourra cependant profiter des classes libres du projet http://www.classpath.org. Cette machine virtuelle implémente l'interface OJI [6], il est donc possible de l'inclure dans http://www.mozilla.org.

La dernière version datant de mars 1999, on est amené à se poser des questions sur le suivi du projet.

Choix d'une machine virtuelle

Au vu de la faible qualité des machines virtuelles libres, on doit se tourner vers les portages de celle de Sun. Pour une machine virtuelle 1.1 ou 1.3, on choisira les versions d'IBM. Pour une version 1.2, on a le

choix entre les portages de Blackdown ou de Sun qui me semblent être équivalents.

Compilateurs Just in Time

Un compilateur Just In Time compile à la volée (pendant que le programme tourne) les classes Java en code natif. Elle accélèrent l'exécution des programmes d'un facteur que l'on peut estimer de l'ordre de 10. Cependant, la compilation prend un temps qui ralentit le lancement de l'application. On aura donc intérêt à désactiver le JIT pour lancer des programmes à durée de vie très courte (comme un compilateur).

Je ne présente ci-dessous que les JITs qui ne sont pas intégrés à des VMs citées ci-dessus.

JIT Borland

- http://www.inprise.com/jbuilder/linux/download/
- Inprise Corporation EXPERIMENTAL TEST SOFTWARE License

Le JIT de Borland est encore en version bêta, mais il semble déjà suffisamment stable pour une utilisation intensive (c'est le JIT recommandé par Sun pour son JDK 1.2.2). Il semblerait que ses performances le placent au niveau des JITs que l'on trouve sur les autres plateformes de développement Java.

TYA

- ftp://gonzalez.cyberus.ca/pub/Linux/java
- Licence http://www.gnu.org/copyleft/gpl.html

TYA est un compilateur Just in Time libre pour les portages du JDK de Blackdown pour architecture Intel. Fonctionne aussi sous BSD. Une machine virtuelle est nécessaire à son fonctionnement.

Sa fiabilité est devenue excellente: l'auteur l'a longtemps utilisé et l'a rarement vu planter. Son fonctionnement est particulièrement simple : il compile une méthode entière en associant à chaque instruction du bytecode une séquence d'instruction en langage machine. *Aucune optimisation* n'est réalisée. Malgré ce fonctionnement simple, les performances sont grandement améliorées (d'un facteur estimé entre 3 et 10 suivant les sources).

Bien que l'accélération des applications soit sensible, ses performances restent très en dessous de celles des JIT de Borland, Sun ou IBM. Son intérêt a donc beaucoup pâti des récents portages des VMs des principaux acteurs du marché Java.

shuJIT

- http://www.shudo.net/jit
- Licence http://www.gnu.org/copyleft/gpl.html

ShuJIT est un compilateur à la volée comparable à TYA. Ses performances s'en approchent et il est distribué sous licence GPL. Il fonctionne avec les portages du JDK 1.1 pour Linux et FreeBSD sur x86.

L'auteur n'a pas eu l'occasion de le tester longuement, mais le version 0.2.10 semble fonctionner correctement avec les programmes testés. L'auteur du programme propose des tests comparatifs sur sa page. Ils tendent à prouver que shuJIT est comparable, en termes de performances, à TYA.

Cacao

- http://www.complang.tuwien.ac.at/java/cacao
- Licence http://www.gnu.org/copyleft/gpl.html

Cacao est un compilateur JIT libre pour Alpha (Linux et Digital UNIX). Non testé (l'auteur n'a pas la chance de disposer d'une station Alpha).

Metrowerks

- http://khendricks.ivey.uwo.ca/linuxppc/
- gratuit

Metrowerks propose un JIT pour mklinux (Power PC) en complément du JDK 1.1.7. Non testé.

Navigateurs compatibles Java

Disposer d'un navigateur compatible Java est indispensable pour développer des applets dans de bonnes conditions car l'applet viewer de Sun n'affiche pas la page HTML dans laquelle tourne l'applet (il n'affiche que les applets du fichier HTML passé en argument).

Netscape Navigator

- http://www.netscape.com
- gratuit

On ne présente plus Netscape et son célèbre navigateur. La version disponible pour Linux souffre cependant d'une stabilité incertaine et d'une machine virtuelle très lente. Sa très large diffusion en fait cependant un outil de test indispensable.

Il est maintenant possible d'installer une bien meilleure machine virtuelle depuis la disponibilité du Java Plugin. Cependant, dans la mesure où ce Plugin doit être installé pour visualiser les applets et le tag utilisé n'étant pas le classique tag <applet>, il n'est intéressant que pour les environnements où l'on contrôle le client (comme en intranet par exemple). on peut alors distribuer des applets utilisant Swing par exemple ou des nouvelles fonctionnalités de la version 1.2 de Java. Un outil permet de traduire les tags <applet> pour une utilisation avec le plugin.

Hot Java

- http://java.sun.com/products/hotjava
- gratuit

HotJava est un navigateur de Sun écrit en Java. Il fonctionne donc sur toute plateforme disposant d'une machine virtuelle (la version actuelle, la 3.0, tourne sur des machines virtuelles 1.1 mais pas sur des VMs 1.2). On notera que Sun propose une version Linux au téléchargement.

Ses performances dépendent bien sûr de celles de la machine virtuelle sur laquelle il tourne. Il est cependant très intéressant pour tester des applets en situation dans une page HTML avec des performances acceptables.

Il existe aussi un composant HTML (JavaBean) permettant d'inclure un navigateur dans ses applications.

Navigateur ICE

- http://193.214.249.162
- commercial, une version de démonstration est disponible.

ICESoft propose un navigateur 100% pur Java, utile pour tester ses applets. Un composant HTML est aussi disponible pour inclure un navigateur dans ses applications.

Compilateurs

Je présente ci-dessous les compilateurs Java autres que l'outil javac présent dans le JDK.

Jikes

- http://www10.software.ibm.com/developerworks/opensource/jikes/project/
- http://www10.software.ibm.com/developerworks/opensource/license10.html

Jikes est un compilateur Java écrit en C et donc extrêmement rapide. Il a été développé par IBM qui l'a pourvu d'une licence de type logiciel libre.

Jikes est très utile pour compiler rapidement de gros projets, mais ses messages d'erreurs sont assez différents de ceux du JDK et donc parfois déroutants. Cependant, il gère mieux les dépendances que javac. Le byte code produit semblerait moins optimisé que celui produit par javac. Une option en ligne de commande (+E) lui permet de formater ses messages d'erreurs de manière à être compris par Emacs.

Jikes est quasiment incontournable pour compiler rapidement de gros projets.

Pizza

- http://www.cis.unisa.edu.au/~pizza
- Gratuit

Pizza est un compilateur Java gratuit. Les sources sont disponibles pour un usage non commercial. Il est écrit en Java, il faut donc une machine virtuelle pour utiliser ce compilateur.

Il présente la particularité d'étendre le langage avec les fonctionnalités suivantes:

1. Polymorphisme paramétrique: permet de paramétrer un type (semblable aux templates du C++).
2. Fonctions de première classe: peuvent être passées comme paramètre, stockées dans des variables et renvoyées par des méthodes.
3. Transtypages permis entre types de base (int, boolean) et objets wrapers (Integer, Boolean).

Ces fonctionnalités sont intéressantes (peut être moins depuis que Java intègre l'introspection), mais le code n'est pas très lisible, et toutes ces fonctionnalités peuvent être implantées de façon classique (mais plus laborieuse). De plus, un programme utilisant ces fonctionnalités (dans des fichiers source dont l'extension est .pizza) ne peut, bien sûr, être compilé que par Pizza.

D'après les tests des auteurs, il compilerait deux fois plus vite que Javac sur des sources volumineux.

Pizza inclut aussi un système de documentation semblable à javadoc.

Choix d'un compilateur

Pour ma part, j'utilise Jikes pour compiler les projets en cours de développement et recompile avec javac pour les tests et la version distribuée.

Compilateurs natifs

Un compilateur natif compile un programme Java (ou des classes Java) en code machine natif. On pourrait s'attendre à un gain de performances substantiel, mais l'expérience montre que l'accélération reste marginale (au plus quelques pourcents) du fait de la grande qualité des JITs actuels. De plus, cette compilation native rend les binaires non portables entre plateformes (bien que le source le reste).

TowerJ

- http://www.towerj.com
- commercial (5000$, 495$ pour un usage non commercial)

TowerJ est un compilateur natif commercial disponible pour de nombreuses plateformes dont Linux. Il permet d'atteindre d'excellentes performances, mais se destine surtout au côté serveur (il n'intègre pas AWT et les libs à distribuer avec l'appli sont assez volumineuses).

Il se classe régulièrement aux premières places dans les tests de machines virtuelles (en particulier au http://www.volano.com/benchmarks.html qui teste les performances côté serveur), cependant, ses performances ne surpassent celles des meilleurs JITs que de quelques pourcents, ce qui ne me semble pas justifier le prix exorbitant de ce compilateur.

GNU Compiler for Java

- http://sourceware.cygnus.com/java
- Licence http://www.gnu.org/copyleft/gpl.html

Cygnus est responsable de l'évolution du compilateur EGCS qui a pris la suite de GCC. Un compilateur natif pour Java est en cours de développement. Des versions bêta sont disponibles sur le site.

La gestion de AWT (graphisme) n'est pas encore implantée, mais cela sera fait dans un avenir proche. Les performances sont encore en dessous de celles des meilleurs JITs, mais cette voie est peut être intéressante sur le long terme dans la mesure où l'on peut espérer que les performances surpassent celles de tous les JITs, notamment en ce qui concerne la vitesse de lancement des applications.

Le sérieux de Cygnus nous laisse entrevoir de bonnes choses pour ce compilateur natif intégré à EGCS (rebaptisé GCC-2.95 depuis que Cygnus est chargé de l'évolution de GCC).

Toba

- http://www.cs.arizona.edu/sumatra/toba
- gratuit

Toba est un compilateur qui transforme des fichiers de classes Java en code source C, permettant ainsi la

construction d'exécutables. On évite ainsi la phase de compilation native des compilateurs JIT.

La mauvaise nouvelle concernant Toba est que le projet a été arrêté, on ne peut donc s'attendre à beaucoup de suivi.

Harissa

- http://www.irisa.fr/compose/harissa
- Licence http://www.gnu.org/copyleft/gpl.html

Harissa convertit le code Java en C et produit un makefile pour compiler le programme. Il comporte aussi un interpréteur. Cependant, il ne supporte que la version 1.0.2 du langage (dont il faut se procurer les classes, soit le fichier `classes.zip`).

On peut se poser des questions sur le suivi du projet dans la mesure où les dernières modifications sur le site remontent à janvier 1999.

La compilation native est-elle un bon choix ?

Les compilateurs natifs peuvent prétendre à de meilleures performances que celles des JITs, mais les implantations actuelles sont soit beaucoup trop chères (TowerJ), soit inefficaces et peu suivies (projets Toba et Harissa).

Cependant, c'est peut être une voie à suivre, seul l'avenir nous le dira.

Environnements de développement

Emacs + JDE

- http://www.gnu.org/software/emacs/emacs.html et http://sunsite.auc.dk/jde
- Licence http://www.gnu.org/copyleft/gpl.html

Emacs est l'éditeur de texte de GNU qui peut être étendu par des fonctions Lisp. On trouve ainsi des extensions pour lire son email, programmer en C, vérifier l'orthographe, etc. Si son aspect est rebutant au premier abord, sa richesse en fait un outil indispensable, et notamment pour la programmation Java.

JDE est un ensemble de fonctions Lisp permettant de tirer le meilleur parti d'Emacs pour la programmation Java. Il utilise les outils (compilateur javac, appletviewer, jdb) du JDK.

Il permet:

1. Par menu de compiler, lancer, débugger, construire une application
2. De naviguer dans les classes (Speedbar)
3. D'afficher la documentation des classes
4. La coloration syntaxique et l'autoindentation
5. D'afficher les lignes de code comportant des erreurs
6. De débugger en affichant le code (avec jdb ou JDebug)
7. De générer automatiquement du code
8. De paramétrer visuellement les outils
9. Intègre l'interpréteur de code Beanshell

Ce programme rend Emacs aussi productif pour la programmation Java qu'un environnement commercial (bien que son aspect soit plus sobre). Ses fonctions se résument à l'essentiel, mais cela me semble largement suffisant. Le tout est extrêmement stable et s'intègre parfaitement à Emacs.

Il existe des extensions bien pratiques à JDE. On notera en particulier:

- http://sunsite.auc.dk/jde/contrib/jdok.el : permet de générer automatiquement le squelette de commentaires JavaDoc, avec paramètres et valeur de retour. Vite indispensable pour commenter ses sources.
- http://sunsite.auc.dk/jde/contrib/javahelp.el : affiche la documentation de Sun par recherche du mot se trouvant sous le point.
- http://sunsite.auc.dk/jde/contrib/jsee.el : construit la documentation javadoc de la classe du buffer et l'affiche dans le navigateur par défaut de Emacs.

Une version bêta (la 2.1.6) intègre un nouveau débuggeur visuel (JDebug) utilisant l'API JPDA.

La grande qualité de JDE est la stabilité et le fait d'être intégré à Emacs (ce qui rend l'utilisation plus simple pour les aficionados de cet éditeur).

On peut lui reprocher l'absence de débuggeur puissant (l'usage de JDB est tout de même assez pénible) et d'un constructeur visuel d'interface. De plus, la configuration peut s'avérer délicate pour un débutant : les options de configurations sont très nombreuses et on a vite fait de se perdre dans les menus.

JBuilder 3.5 Foundation

- http://www.borland.fr/Download/jbuilderfondation/index.asp
- Gratuit en version Foundation

L'investissement de Borland dans Linux s'est concrétisé ces derniers mois avec la diffusion d'un JIT et de leur EDI Java JBuilder.

La version Foundation est la nouvelle génération de leur EDI écrite 100% en Java, ce qui du coup la rend disponible pour toutes les plateformes bénéficiant d'une machine virtuelle performante (versions Windows, Solaris et Linux). Cette application est intéressante à plus d'un titre :

- Elle montre qu'une application de grande diffusion peut être réalisée en Java (avec les JFC)
- Elle démontre la portabilité de Java dont certains doutaient (*write once, test everywhere*), surtout du fait des problèmes supposés de portabilité des applications AWT.

Non seulement cette application est novatrice, mais elle est très agréable à utiliser. On remarquera en particulier sa stabilité qui n'est pas sans rappeler celle des produits de la meilleure époque de Borland (comme Turbo Pascal ou Borland C++). On y trouve tout ce que l'on peut attendre d'un EDI de qualité : un éditeur de bonne facture, un excellent débuggeur et quelques petits plus qui facilitent la vie du développeur, comme le rappel des arguments d'une méthode ou la vérification syntaxique en temps réel. Le constructeur visuel d'interface est agréable à utiliser et produit un code de qualité, mais que l'on peut cependant optimiser à la main.

Cette version Foundation est modulaire et peut accueillir des extensions. Borland fournit une API appelée OpenTools, pour le développement de telles extensions. Pour plus de renseignements sur cet aspect de JBuilder, on pourra visiter la page de Blake Stone, l'architecte de JBuilder, qui détaille cette architecture et propose des tutoriels à l'adresse http://homepages.borland.com/bstone.

JBuilder 3.5 existe en version Foundation (sans limitations de durée mais ne disposant pas de tous les outils des autres versions), d'une version professionnelle intégrant la gestion des bases de données et la version entreprise gérant les EJB [7]

On pourra reprocher à JBuilder l'absence de gestionnaire de version et des limitations parfois sévères dans la version Foundation (il est ainsi impossible de générer une interface, il faut générer une classe puis modifier le source *à la main* pour en faire une interface). De plus, il faut disposer de beaucoup de RAM pour être à l'aise (compter 128 Mo).

L'outil de conception visuelle d'interface se veut bidirectionnel (on devrait pouvoir éditer le code produit par le logiciel), mais mon expérience montre que c'est à éviter : il peut arriver que l'on perde du code que l'on a édité si l'on retourne au module de conception visuelle.

Malgré ces petits défauts, JBuilder reste un excellent outil qui a permis à Borland de gagner une place de leader dans le marché des outils de développement Java.

Forte for Java

- http://www.sun.com/forte/ffj/
- Gratuit en version Community

Sun a récemment fait l'acquisition de NetBeans et de Forte, deux fournisseurs d'environnements de développement Java pour remplacer ses outils Java Workshop et Java Studio. Forte for Java est disponible aussi en version Internet (qui intègre les outils de la plateforme entreprise de Java, donc les EJB, RMI, Corba et JNDI) et Entreprise (qui permet le travail en équipe et facilite le déploiement des applications).

NetBeans a été le premier outil de développement Java d'envergure codé en Java et disposant de possibilités d'extension. Il sera par exemple possible d'y intégrer Together/J (outil de modélisation UML et de génération de code) dans une prochaine version.

Forte for Java intègre un module visuel de conception d'interface graphique, un débuggeur un système de complétion de code et facilite le développement des JSP [8]. De plus, il comporte un système de mise à jour automatique par l'internet (il se connecte régulièrement au site de Sun et propose de télécharger de nouvelles extensions ou des mises à jour lorsqu'elles sont disponibles).

On reprochera à Forte for Java son appétit de mémoire et de cycles CPU qui le rend inutilisable sur de petites configurations (128 Mo de RAM est un minimum).

Visual Age for Java

- http://www-4.ibm.com/software/ad/vajava/
- Gratuit en version Entry

Visual Age for Java est l'environnement de développement Java d'IBM. Vu l'investissement de ce dernier dans la technologie Java, on pouvait s'attendre à un bon outil, et on est pas déçu !

IBM propose une version "Entry" gratuite et sans limitation, si ce n'est le nombre de classes limité à 750 dans le repository. IBM propose aussi une version standard sans limitations et une version Entreprise permettant le travail en équipe (gestion commune du repository).

VAJ est un outil trés performant mais difficile à prendre en main car déroutant : le code est inaccessible dans sa globalité et on ne peut y accéder que par méthodes. Il est cependant possible d'exporter le code sous forme de sources ou de classe Java. Le code est entreposé dans un repository (comparable à celui de CVS) et entièrement géré par le système. Lorsqu'on enregistre une méthode, un paquet ou un projet, Visual Age réalise une compilation incrémentale et signale les erreurs éventuelles. Toutes les versions d'une méthode sont accessibles et l'on peut *geler* une version (en lui donnant un nom) pour un projet, un paquet ou une méthode. Cette version sera ensuite toujours accessible.

Hormis ces particularités, Visual Age propose tout ce que l'on peut attendre d'un EDI de qualité : débuggeur, construction visuelle de l'interface (et de la gestion des événements).

On aura compris que Visual Age for Java est un outil très novateur, mais dont la prise en main est difficile (mais la documentation fournie est complète, compter tout de même une semaine pour faire le tour des fonctionnalités de l'outil). Ces qualités le rendent cependant très dépendant des outils d'IBM et il n'est pas possible de changer de machine virtuelle. La version actuelle (la 3.0 intégrant une machine virtuelle 1.1.6) commence à accuser son âge. En particulier, il est impossible de travailler dans de bonnes conditions avec la version 1.1.1 des JFC, version compatible avec les versions postérieures du JDK. La sortie d'une machine virtuelle 1.3 pour Linux laisse cependant supposer qu'une nouvelle version de Visual Age est imminente.

Autres EDIs disponibles sous Linux

Il existe nombre d'autres EDIs disponibles sous Linux. En particulier :

- http://www.omnicore.com/ : C'est un IDE très classique mais de bonne qualité. Il a été programmé entièrement en Java mais est relativement peu gourmand en ressources. Il comporte un bon débuggeur et un éditeur permettant la complétion du code. Cela peut être une alternative intéressante pour des machines peu puissantes, mais son prix est dissuasif lorsqu'on compare ses fonctionnalités avec la version gratuite d'IDEs comme JBuilder ou Forte.
- http://www.elixir.com.sg : Elixir est aussi un environnement de développement commercial écrit en Java, mais il fait un usage raisonné de Swing, ce qui le rend rapide même sur des configurations modestes. Il reste simple mais très fonctionnel. Il permet de gérer le système de contrôle de version RCS et intègre un système de scripts Scheme permettant d'implémenter ses propres fonctions.
- http://www.freebuilder.org : Projet d'EDI libre, FreeBuilder est une véritable légende urbaine, dont tout le monde parle, mais que personne n'a jamais utilisé (ni même réussi à compiler). Le projet a longtemps été arrêté, mais il semble avoir été relancé depuis peu sous le nom évocateur de FenIX. On ne peut que souhaiter qu'il reparte du bon pied et concurrence sur leur propre terrain les EDIs commerciaux.

Outils divers

Argo UML

- http://www.ArgoUML.org/
- Licence Open Source

Argo UML est un outil de modélisation UML Open Source. Il permet de produire des diagrammes UML et le code Java correspondant. Bien que non terminé, il est suffisamment avancé pour être utilisable. On notera qu'il ne permet pas (encore ?) l'analyse inverse (donc production de diagrammes à partir de code).

C'est certainement une alternative viable face aux outils de modélisation UML commerciaux souvent très chers.

TogetherJ

- http://www.oisoft.com
- commercial

TogetherJ est un outil commercial de modélisation UML capable de tracer les graphes à partir de code Java ou de générer le code à partir des graphes. Une version d'évaluation assez limitée (elle ne permet que de produire des diagrammes de classes et l'on ne peut exporter le résultat) est disponible gratuitement.

C'est un très bon outil, mais son prix élevé peut en dissuader plus d'un.

Alma

- http://www.memoire.com/guillaume-desnoix/alma
- Licence http://www.gnu.org/copyleft/gpl.html

Alma est un logiciel disposant des fonctionnalités suivantes:

1. Lecture et analyse de code-source écrits dans divers langages.
2. Manipulation de la structure et du code
3. Génération de code-source dans divers langages.

Il se destine à la modélisation orientée objet (définition de classes et relations) ainsi qu'à la migration (aide à la conversion) de code écrit dans des langages plus vieux. Il correspond à deux besoins : disposer d'un AGL simplifié et utilisable sur de petits projets et faciliter la réécriture, le portage ou l'encapsulation de code non-objet.

Ce logiciel servira principalement au développeur qui récupère un code et souhaite l'intégrer dans un autre projet, à porter un code standard sur la machine virtuelle Java, ainsi qu'au concepteur qui y trouvera (à terme) la possibilité de déclarer et manipuler des classes.

Ce programme est très bien réalisé (il y a même un assistant !), et tourne très correctement, même sur une configuration modeste.

Optimize-it !

- http://www.optimizeit.com
- commercial

Optimize it! est un optimisateur de code Java. Il permet de visualiser, pendant qu'une application Java s'exécute, l'occupation mémoire pour chaque classe ou le temps d'exécution pour chaque méthode. Sa version 3.0 est maintenant disponible pour Linux (ainsi que pour Windows et Solaris).

C'est un bon outil pour optimiser ses applications Java en travaillant les goulots d'étranglement du code.

JAD

- http://www.geocities.com/SiliconValley/Bridge/8617/jad.html
- http://www.gnu.org/copyleft/gpl.html

C'est un désassembleur, donc un outil capable de générer un source à partir d'un fichier .class. C'est utile lorsque l'on a perdu le source de ses propres classes...

JAX

- http://www.alphaworks.ibm.com/formula/Jax
- gratuit

Jax est un outil d'optimisation du bytecode Java. Il permet ainsi d'optimiser les performances des programmes, mais surtout de réduire leur taille. On peut aussi l'utiliser pour camoufler le code source (c'est donc un antidote aux désassembleurs).

Ant

- http://jakarta.apache.org/ant
- Licence Apache

Ant est l'équivalent d'un make pour les programmes Java. En effet, make appelle des programmes du système hôte qui ne sont pas présents sur toutes les plateformes, les makefiles ne sont donc pas portables. Pour que les projets Java soient recompilables sur toute plateforme, il faut donc disposer d'un outil (Java) de compilation des projets.

Un buildfile Ant est un fichier XML (format de fichiers textes structurés et portables) indiquant à Ant les étapes de la construction du programme.

```
<project name="foo" default="dist" basedir=".">

  <target name="init">
    <tstamp/>
    <property name="build" value="build" />
    <property name="dist"  value="dist" />
    <filter token="version" value="1.0.3" />
    <filter token="year" value="2000" />
  </target>

  <target name="prepare" depends="init">
   <mkdir dir="${build}" />
  </target>

  <target name="compile" depends="prepare">
    <javac srcdir="${src}" destdir="${build}" filtering="on"/>
  </target>

  <target name="dist" depends="compile">
    <mkdir dir="${dist}/lib" />
    <jar jarfile="${dist}/lib/foo${DSTAMP}.jar"
    basedir="${build}" items="com"/>
  </target>

  <target name="clean" depends="init">
    <deltree dir="${build}" />
```

```
    <deltree dir="${dist}" />
  </target>
</project>
```

Ant est un projet de l'organisation jakarta.apache.org ayant pour mission de fournir des outils Java côté serveur de qualité. Il est utilisé pour construire des outils comme Tomcat ou Watchdog.

Conclusion

La situation de Java sur plateforme Linux a radicalement changé ces derniers mois. De plateforme de développement Java de second ordre, Linux est passé dans le peloton de tête grâce à l'implication des acteurs du marché. On dispose maintenant de machines virtuelles récentes et de qualité et les environnements de développement majeurs ont tous été portés (sauf un dont le développement a par ailleurs été arrêté... mais dont on arrive à se passer).

Le développeur Java n'a maintenant plus rien à envier à ses homologues travaillant sur d'autres plateformes.

Notes

[1] Cette limite est de 2048 processus par utilisateur. On peut modifier cette limite en éditant le fichier include/linux/tasks.h de l'arborescence des sources du noyau. La constante NR_TASKS indique le nombre maximum de processus dans le noyau et MAX_TASKS_PER_USER le nombre de processus par utilisateur, sa valeur par défaut est NR_TASKS/2. En modifiant les valeurs de ces constantes, on peut choisir la limite désirée.

[2] Le JRE est le minimum nécessaire pour faire tourner une application Java : il ne comporte qu'une VM et tous les outils de développement en ont été éliminés pour l'alléger au maximum. Compter tout de même une dizaine de Mo.

[3] Le Java PlugIn permet d'ajouter une machine virtuelle Sun à Netscape. Cela est très utile vu la qualité très médiocre de la VM livrée avec les versions 4 du Navigator. Il faut cependant changer les tags <applet> en <embed>. Sun propose un outil pour réaliser automatiquement cette conversion.

[4] JPDA (pour Java Platform Debugger Architecture) est un support de débuggage pour la plateforme Java 2. Il définit des APIs utilisées par les débuggeurs modernes pour accéder aux informations de débuggage.

[5] Pour des applications ne durant que quelques secondes, on aura intérêt à désactiver le JIT en donnant à la variable d'environnement JAVA_COMPILER une valeur quelconque.

[6] OJI pour Open Java Interface.

[7] Entreprise Java Beans, des composants métier à intégrer à des serveurs d'applications.

[8] Java Server Pages, ce sont des pages HTML comportant du code Java et compilées automatiquement sous forme de servlets par le serveur lui même.

Dernière mise à jour : 2000-05-12