Developpez Magazine

Hors Série spécial Java – Javapolis 2006.
HS-Java-0
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Cédric Chatelain
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Index

Editorial

Cette cinquième édition de Javapolis à Anvers attend près de 3000 personnes et accueillera de prestigieux speakers.
Elle va aussi clôturer une année riche en évènements pour l'univers Java.
A l'occasion de cette édition de Javapolis, **www.developpez.com** vous propose une édition spéciale Java de son journal, pour vous faire découvrir quelques unes des ressources que vous pourrez retrouver sur la rubrique Java du site **java.developpez.com**
Alors bonne lecture, bon Javapolis et à bientôt sur les forums de www.developpez.com

La rédaction Java

Tutoriel JAVA

Le garbage collector

Le Garbage Collector, que je nommerai par la suite GC pour ne fatiguer ni vos yeux ni mes doigts, est une sentinelle bienveillante présente dans toutes les machines virtuelles. Son rôle consiste à identifier puis à libérer les zones de mémoires inutilisées par l'application en cours d'exemple.

par **Romain Guy**
Page 12

Blog JAVA

Comprendre les Références en Java

*L'API Java est vraiment énorme. Elle offre un grand nombre d'outils et de possibilités qui, bien souvent, ne sont pas utilisées tout simplement parce qu'elles sont méconnues... Cela faisait quelques temps que je voulais m'intéresser plus particulièrement aux diverses références de **Java***

par **adiGuba**
Page 15

Evènements

Janvier

- sortie de Sun Java Studio Creator 2

Février

- sortie de NetBeans 5

Avril

- Codecamp a Paris

Mai

- Java EE 5
- La sortie du Google Web Toolkit
- JavaOne avec son lot d'annonce :
  - le passage en open source de Java
  - l'annonce d'un JBuilder 2007 basé sur Eclipse
  - la discussion autour des closures qui continue toujours
  - l'annonce de la licence DLJ permettant une distribution avec Linux
- JAX 2006 en Allemagne

Juin

- la sortie de Callisto (synchronisation des 10 principaux projets de la fondation Eclipse).
- la venue de James Gosling a Paris pour JavaDay

Octobre

- la sortie de la versions 5.5 de NetBeans, ses 8 ans et la preview technique de VWP
- Sortie de IntelliJ Idea 6.0
- le projet de traduction de Javadoc de James Gosling
- le Java Module System pour Java SE 7 en early draft
- la version finale de Spring 2.0
- Eclipse Summit Europe en Allemagne et Eclipse - now you can à Paris

Novembre

- l'annonce de la licence de Java : GPL v2
- Duke en open Source
- les 5 ans d'Eclipse

Enfin, à venir à l'heure où nous écrivons ces lignes : Java SE 6

Javapolis 2006

L'évènement Java de fin d'année

Interview exclusif de Stephan Janssen, fondateur de Javapolis

1. Présentation

1.1. Stephan, pourrais tu te présenter à nous tous: qui tu es ? pour qui travailles tu ? tes passions ? ...

Je développe depuis l'âge de 14 ans, je hackais.. euh étudiait les jeux sur Commodore 64 et j'écrivais des introductions et éventuellement des démos en assembleur. Après une période de LU6.2 SNA communication development je suis passé à Java en 1996. Nous avons utilisé JDK 1.0.2 dans une compagnie de logiciel financier, appelée FICS. La j'étais impliqué dans les différentes couches de la solution, en commençant par des applets et AWT, une connection par socket sécurisée, pour terminer par un serveur Java. Ca pouvait évoluer vers des solutions de banking en utilisant Swing, RMI, et J2EE.En 1998, j'ai fondé une compagnie de consultance Java, appelée JCS(Lien1). En 2004, j'ai été acquis par le Group Dolmen où nous avons une équipe de plus de 100 développeurs et architechte Java faisant des projets Entreprise Java, utilisant principalement Spring & Hibernate/JPA et maintenant récemment Java EE 5.0.

Ainsi, en journée je suis le CTO de JCS et le soir et les week-ends je suis chairman de BeJUG/JavaPolis.

2. A propos de BEJUG ?

2.1. Qu'est ce que le BEJUG ?

BEJUG est le Java User Group pour la Belgique, dont je suis le fondateur. Nous organisons plusieurs fois par an des conférences (la dernière était une conférence consacrée à SOA, et s'est tenue en Octobre), mais aussi des événements comme Spring et Javapolis.

2.2. Qu'est ce qui t'a poussé à créer le BEJUG ?

En 1997, je désirais rencontrer d'autres développeurs Java. J'ai donc contacté Sun pour un serveur et le jour suivant BeJUG était né :)

3. A propos de Javapolis

3.1. Qu'est ce qui t'a poussé à créer Javapolis ?

Chaque année, plusieurs personnes de JCS (moi y compris) se rendaient à JavaOne. Mais lorsqu'on regarde l'investissement pour s'y rendre et le manque d'alternative en Europe, j'ai commencé une conférence appelée JavaPolis(Lien2) avec le BeJUG en 2002 avec une mission claire: avoir une alternative européenne et bon marcher pour JavaOne!

3.2. En a-t-on vraiment pour son argent ?

JavaPolis est constituée de 2 parties: 2 jours d'université (sessions techniques en profondeur) et 3 jours de conférences (des sessions d'1 heure couvrant différents sujets de Java). Durant ces parties, il y a également les BOFs (en soirée), les Quickies (durant la pause de midi), les LABs (sessions pratiques) et des réunions informelles organisées par différentes personnes devant des tableaux blancs.

3.3. Concernant le thème de cette année (there are better ways to meet your idole), ou vas-tu chercher ces idées ?

Nous travaillons avec plusieurs compagnies de communication et marketing qui nous suggèrent différentes idées basées sur la direction que nous leur donnons. Notre public cible pour Javapolis est (malheureusement) principalement masculin (95%) entre 20 et 40 ans... ce qui explique probablement les différents thèmes Javapolis du passé ;)

3.4. Quelle est ton(tes) idole(s) cette année ?

Cette année, nous avons beaucoup d'idoles Java qui seront présents à JavaPolis. Mes idoles favorites pour cette année sont Erich Gamma (GoF et Eclipse), Neal Gafter (qui nous parlera des *closures* en Java) et dans un style spécial Marc Fleury :)

3.5 Les coulisses de la vidéo:

3.5.1. Est-ce vraiment James Gosling qu'on voit sur la vidéo ?

Je suis très heureux que vous me posiez cette question. Près de 10.000 personnes ont vu le clip, et lorsqu'on voit les réactions, la plupart pensent que c'est James Gosling...
Et bien, je vais te dire un secret: la personnes dans la vidéo est un acteur belge ressemblant à James Gosling qui a étudié plusieurs films de James pour l'imiter aussi bien que possible. Cependant, le vrai James Gosling a donné son accord pour ce film, ce qui montre le bon sens d'humour qu'il possède!

3.5,2. Certains on cru voir un livre .net dans sa poubelle. Un clin d'oeil ?

C'était pour nous faire plaisir ;)

3,6, Le choix des Keynotes

3.6.1. Pourquoi avoir choisi Marc Fleury ? Un rapport avec le fait que Java sera rendu Open Source cette année ?

Les années précédentes, "il n'était pas possible" d'avoir Marc Fleury pour le discours d'ouverture à JavaPolis. Au lieu de cela, il fréquentait le lieu incognito, comme Zorro. Cette années, avec l'acquisition de Red Hat, et le fait que JBoss soit un des principaux sponsors, il était clair pour moi que Marc devait faire le discours d'ouverture.

3.6.2. L'année passée, la vedette du salon, c'était SOA. On en parlait à tous les stands. Et cette année, la vedette, ce sera qui ?

Probablement très similaire à JavaOne qui fut AJAX, Langages Dynamiques, et SOA.
Concernant AJAX, nous avons le fondateur de DWR et Dojo comme orateur, et nous aurons quelqu'un de Google pour nous parler de GWT.
Les fondateurs de JRuby, maintenant employés chez Sun, seront présents à l'université et feront une conférence.
Les développeurs du Projet Simplice (Visual Basic pour Java) seront également présents.
Concernant SOA, nous avons Guy Crets et Robin Mulkers qui ont sélectionné différents sujets SOA et des orateurs comme alternative au *IDEs en Action* de l'année passée. Cela devrait être bien!
Bien sûr les sessions JavaPolis sur JDK 6 et 7 et l'open sourcing du langage Java va certainement attirer pas mal d'attention cette année également.

3.7. 3000 personnes, ce n'est pas un peu trop pour les lieux ? On ne va pas se marcher sur les pieds ?

Je pense que nous devrions être entre 2,500 et 3,000 personnes cette année. Pour anticiper ce volume, nous avons fait quelques aménagements par rapport aux autres années:

- ajout d' une salle supplémentaire (5 salles au total)
- projection des Keynotes également dans la 2ième grande salle
- 80 mètres de tables seront rajoutés au niveau supérieur pour que les personnes puissent se relaxer, vérifier leurs e-mails, ou recharger leur portables.
- Finalement, nous allons créer une zone Salon dans le hall près de la zone d'exposition où les personnes auront des tableaux blancs et de l'espace pour se relaxer

J'espère que cela suffira :-)

3.8. Y aura-t-il une connection Wi-Fi pour tout le monde ?

L'année passée, nous avons eu de sérieux problèmes avec le fournisseur d'accès 'international'. Aussi, nous avons décidé de le faire nous-même. Nous avons testé notre configuration Wi-Fi durant SpringOne dans le hall d'exhibition et les salles, et tout à fonctionné parfaitement. Donc OUI, tout le monde aura un connection wifi *GRATUITE* durant JavaPolis.

3.9. La tradition veut qu'il y ait une séance de cinéma le jeudi soir. Quel sera le film cette année ?

Nous venons juste de lancer le vote pour le film Javapolis (http://www.javapolis.com/confluence/display/JP06/Movie) Vous pouvez choisir parmi les films suivants:

- Flushed away
- Casino Royal
- Talladega Nights
- Happy Feet

Le film ayant le plus de votes sera montré jeudi soir.

3.10. Difficile d'organiser un tel événement ?

Cette année, c'est notre 5ième édition. Donc je me sens de plus en plus à l'aise dans l'organisation de Javapolis, à différents niveaux.
L'inscription en ligne est vraiment ce que je voulais (excepté pour le payement en ligne) grâce à Frédéric.
Les relations avec nos partenaires se font à la vitesse maximale, et du point de vue contenu, il est plus facile d'avoir des orateurs, comparé à nos débuts.
Notre wiki s'améliore également chaque année aussi bien du point de vue navigation que du point de vue Look and Feel.
Néanmoins, cela reste une tâche énorme et les préparations commencent 8 mois avant l'événement, en fait!!

3.11. Où trouver cette énergie pour organiser cet événement ?

Accueillir plus de 2.100 développeurs Java depuis plus de 45 pays me donne assez d'énergie pour continuer chaque année avec JavaPolis!
J'aime également les défis, avoir de nouvelles idées pour améliorer l'expérience de conférence en général, comme notre DVD ou des présentations en-ligne, pousser le wiki à un niveau plus élevé, etc.

3.12. Comment faire pour que cela reste un plaisir ?

Trouver des sujets Java intéressants et de bons orateurs lors de différentes conférences Java, PARTOUT dans le monde est vraiment quelque chose que j'aime faire.
Aussi, depuis le mois de janvier 2006, j'ai engagé un coordinateur d'événement à plein temps qui m'aide pour la logistique, les DVD, les newsletters, etc, ce qui me donne plus de temps pour d'autres initiatives plus plaisantes.

3.13. Comment se fait la sélection des orateurs ?

Nous avons un steering committee à Javapolis où chaque personne est responsable d'un Track donné. Cette personne peut suggérer des sujets et des orateurs que nous discutons lors de nos réunions mensuelles.
Du fait que nous sommes tous des volontaires, JavaPolis est vraiment NOTRE conférence avec des sujets qui nous intéressent et des orateurs que nous aimerions voir. C'est pourquoi c'est une conférence où l'on pioche, et jusqu'à présent, cela a toujours fonctionner.

3.14. Pas trop de pression de la part des partenaires ?

Je travaille avec une compagnie externe qui fait la "recherche de

fonds" et les partenariats. Cette compagnie travaille donc comme un buffer pour moi, me donnant à nouveau plus de temps pour faire d'"autres" choses :)

3.15. Comment réagit ta femme vis-à-vis de ta vie très chargée ?

Ma femme préfère que je sois assis en face de mon ordinateur plutôt qu'accroché à un bar à boire de la bière toutes les nuits :o-
Aussi, je l'ai impliquée dans l'administration.

3.16. Y a-t-il plus d'inscriptions provenant de France cette année ?

Je n'ai pas encore regardé les détails, mais j'espère que des articles comme celui-ci va convaincre les personnes de France de se rendre à JavaPolis. Nous avons délibérément rajouter la tour eifel dans le logo de Javapolis, en espérant démontrer que les développeurs Francais sont également les bienvenus :)

3.17. Comment expliques tu que la France était absente l'année passé (moins de 50 personnes, si je me rappelle bien) ?

Je ne sais pas. Peut-être la barrière du langage ? Parce que d'un point de vue de déplacement, c'est juste la porte à cotée avec le Thalys !

4. Concernant Java

4.1. Quels sont les plus du langage Java, selon toi ?

Le fait que Java est un langage OO ouvert et indépendant de la plateforme est juste très puissant. Tu peux également apprendre de nombreuses API (standards), comme récemment JPA, et déployer ces librairies dans n'importe quel projet IT est un énorme bénéfice pour tous les développeurs Java.

4.2. Quels sont ses faiblesses ?

Pour le moment, avec J2EE, la complexité ne doit pas être sous-estimée. Cependant, avec des frameworks comme Spring et le nouveau Java EE 5, la facilité de développement et la simplicité sont clairement les buts de ces mouvements autour de l'Entreprise Java.

4.3.. Quel est ton EDI préféré ?

J'utilise IntelliJ(Lien3) sous Mac OS X

4.4. Quel est ton serveur J2EE préféré ?

Avec Spring(Lien4) et Hibernate(Lien5) , Je n'ai besoin que de Resin (de Caucho(Lien6)) mais autrement WLS(Lien7) ou Jboss(Lien8).

4.5. Quel est ton API préférée ?

JMS(Lien9)

4.6. Java va être rendu open-source. Une bonne chose selon toi ?

Pour moi, Java était déjà assez ouvert. Mais je comprends que lorsqu'un développeurs désire être impliqué dans la résolution des bogues ou l'extension du langage, que le model actuel ne suffisait pas. Cela sera un grand thème durant JavaPolis, où Sun donnera plus d'informations durant leur KeyNote et donnera plus de détails durant une session BOF.

4.7. Est-il possible de faire du business autour de Java ?

En 98 JCS faisait principalement de la consultance Java dans les pays scandinaves.
Ces jours-ci, avec Dolmen/JCS nous avons plus de 100 développeurs Java et chacun d'eux travaille sur des projets Java en Belgique. Donc, oui. Java est un GROS business!

4.8. Tu es également un Java Champion. Que penses-tu de ce programme, de pouvoir rencontrer et découvrir d'autres Java Champions ?

C'est super d'être reconnu pour les efforts que nous faisons pour l'éco-système Java. C'est également un façon agréable de rencontrer d'autres passionnés de Java, et de s'échanger les idées et les vues ! J'aime beaucoup !

Merci beaucoup Stephan pour cet interview, et nous te souhaitons beaucoup de succès pour Javapolis.

Cet interview a été réalisé début novembre 2006

Le choix de la rédaction

Lundi
Dynamic Languages on the java platform: une présentation de la JSR 223 ainsi que 2 langages populaires que sont Groovy et Jruby

DOJO in action: simplifier le développement Web

Mardi
RIA using Swing: vous montrera ce qu'il est possible de faire aujourd'hui avec Swing, ainsi que quelques bonnes pratiques.

Java Performances Tuning: comment booster vos applications

Mercredi
OpenJDK: BOF du soir expliquant tout sur OpenJDK, la version open source de l'implémentation de Sun de Java SE

The Java SE Platform – Past and future: Comment la plateforme Java SE va encore évoluer dans le future.

Jeudi
Java Specialist in Action: présenté par un Java Champion, vous apprendrez comment écrire du code hautement flexible

Project Semplice: La puissance de Java à portée des développeurs VB

Vendredi
OpenOffice & JDK: Comment manipuler des documents OpenOffice depuis Java

Automatic Testing using open source tools : automatisez vos tests sans dépenser de fortunes en licence

Et surtout, ne ratez pas les keynotes du mercredi et jeudi matin

Java rendu Open Source

On retrouvait Java dans tous les secteurs d'activités: Dans nos PC de bureaux, dans les serveurs, dans les téléphones mobiles, les assistants numériques, les caméscopes, les décodeurs TV, à la NASA, dans nos cartes d'identités (du moins, en Belgique)
On avait l'impression que tous les domaines étaient occupés par Java ? Tous ? Non ! Car la FSF résistait encore et toujours à Java. Indiquant que c'était un piège pour tous ceux qui développaient des applications libres.
Et bien, maintenant, Java vient de conquérir ce dernier domaine. Car depuis la mi-novembre Java est disponible sous licence GPL(v2).

Java SE

Java SE est disponible sous licence GPLv2 avec une exception pour le classpath. Bien évidemment, Java SE continuera également d'être disponibles sous les licences commerciales actuelles.
Aujourd'hui, ce qui est disponible, c'est

- le compilateur Java (javac)
- l'utilitaire JavaHelp
- la technologie HotSpot

Vous pouvez télécharger les sources depuis un repository SubVersion, ou pour ce qui est du compilateur java, directement depuis le centre de mise à jour de NetBeans.

Le projet Open Source de Java SE se nomme openjdk, et est hébergé sur java.net
http://openjdk.dev.java.net
https://javahelp.dev.java.net/

Java ME

Tout comme Java SE, Java ME est également disponible sous licence GPLv2. Mais contrairement à Java SE, elle ne possède pas d'exception pour ce qui est du classpath. Sun justifie ce choix par le fait que le runtime de Java ME est intégré dans le matériel et n'a donc pas besoin de cette exception, contrairement à Java SE qui lui n'est pas intégré dans le matériel.

Le projet Open Source de Java Me se nomme phoneme, et est hébergé sur java.net
http://phoneme.dev.java.net

Java EE

Glassfish, l'implémentation de référence de Java EE, supportée par Sun et d'autres acteurs comme Oracle, a également été mise sous licence GPLv2. En plus de sa licence CDDL.
C'est la seule des 3 implémentations qui est donc disponible sous une double licence open source. Cela a pour effet que le code donné par Oracle (le coeur de Toplink) est maintenant également sous licence GPLv2.
http://glassfish.dev.java.net

Duke

Mais Sun n'a pas libéré que Java. Sun a également libéré la célèbre mascotte Duke. Ce qui veut dire que nous pouvons maintenant utiliser librement Duke et ses images. Sun y a même consacré un projet où vous pouvez en télécharger quelques unes:
http://duke.dev.java.net

En résumé

- Java SE, Java ME et Java EE sont maintenant disponible sous licences GPLv2
- Sun a rajouté une Classpath Exception, comme la licence le lui permet, permettant à quiconque de faire tourner ses programmes Java sous la licence de leur choix
- Ils continueront également d'être disponibles sous la licence commerciale actuelle
- Glassfish sera sous licence GPLv2 ET CDDL
- Sont aujourd'hui disponible en open source:
    - la technologie HOTSPOT,
    - le compilateur Java,
    - et l'utilitaire JavaHelp
- Sun espère fournir Un JDK "buildable" pour le 1er trimestre 2007
- Le projet pour Java SE s'appelle OpenJDK
- Le projet pour Java ME s'appelle
- Il faudra obligatoirement passer le TCK pour pouvoir utiliser le nom Java
- Le TCK n'est pas sous licence Open Source
- Le JCP reste en place. Et continuera d'être utilisé pour rajouter des fonctionnalités aux plateformes Java
- il faudra signer une Sun Contribution Agreement si on désire que le code soit intégré dans le JDK/JRE
- Le code source sera tout d'abord disponible en lecture via Subversion, plus tard via Mercurial
- Le code source est disponible sous la forme de projets NetBeans.
- 2 adresses à retenir:
    - http://sun.com/opensource/java
    - http://nb-openjdk.netbeans.org

Retrouvez tous les détails de l'annonce faite par Sun Lien10
Interview de Tom Marble: Lien11
Interview d'Alexis Moussine-Pouchkine: Lien12
Interview de Romain Guy: Lien13
Interview de Vincent Brabant: Lien14

Java SE 6 disponible

La prochaine version de Java ne devrait plus tarder à être disponible en version finale. Il est donc intéressant de prendre le temps de découvrir ce que nous apportera ce nouvel opus. Outre un nouveau changement de nom et de numérotation, ce qui devient presque une tradition, puisqu'on est passé de **J2SE 1.4.2** à **J2SE 5.0** et que l'on a désormais droit à **Java SE 6** (on se demande ce qu'ils nous préparent pour **Java 7** !), cette nouvelle version a mis le focus sur six grands thèmes principaux :

Compatibilité ascendante

Bien que cela puisse paraître évident pour beaucoup de monde, l'accent est mis sur la compatibilité ascendante. Bien que la plateforme soit mature, elle continue d'évoluer et continuera d'évoluer sans cesse, mais cela ne remet nullement en cause la compatibilité des programmes : N'importe quel programme qui fonctionnait sur une version précédente de la plateforme Java devrait fonctionner de la même manière avec Java SE 6.

Simplifier le développement

Java 5.0 a apporté des améliorations significatives avec l'introduction de nouvelles syntaxes, comme les types paramétrés (Generics), les annotations, les énumérations et les autres fonctions du langage. Il reste toutefois encore beaucoup de travail pour simplifier la vie du développeur, en particulier en ce qui concerne l'accès aux bases de données, au support des langages de script, des technologies fondamentales tels que la compilation ou le traitement des annotations ou encore un meilleur support des EDI et des outils élaborés de design d'interface graphique.

Diagnostic, surveillance et gestion

La plateforme Java est utilisée en production pour des applications critiques. **Java 5.0** avait apporté de nombreux progrès avec l'introduction de **JMX** (Java Management Extensions) et **JVMTI** (Java Virtual Machine Tools Interface) mais d'autres améliorations sont cependant nécessaires dans ce domaine.

Le retour du "Desktop"

Les applications riches sont revenues sur le devant de la scène et les développeurs commencent à atteindre les limites des clients légers basés sur les navigateurs. Java est une alternative possible, mais elle souffre toujours de problèmes d'intégrations au sein du système de l'utilisateur final et certains composants souffrent de quelques gros défauts qui devront être comblés.

XML et Web Services

Java 5.0 devait originellement proposer un ensemble d'outils facilitant la création et l'utilisation des Web Services. Malheureusement cela a pris plus de temps que prévu et cela n'a pas pus être intégré dans la spécification, et pendant ce temps XML et les Web Services ont pris de plus en plus d'importance pour beaucoup de membres de la communauté.

Transparence dans l'évolution

La communauté a exprimé son désir de transparence dans l'évolution de la plateforme Java. Beaucoup voulaient pouvoir passer en revue et tester ces nouvelles spécifications alors même qu'ils étaient toujours en progrès, et devenir plus impliqué en contribuant des améliorations ou des corrections de bugs. La processus de développement de cette nouvelle version a donc été un des plus ouverts via son site communautaire : https://jdk6.dev.java.net/

Quoi de neuf docteur ?

Cette nouvel opus de Java nous proposera les nouvelles spécifications suivantes :

JSR 223 : Scripting for the Java Platform

Cette JSR définit un framework permettant d'utiliser n'importe quel langage de script au sein d'une application Java. Elle permet ainsi de connaître la liste des interpréteurs disponible sur la machine virtuelle, d'invoquer des scripts depuis une application Java (et vice-versa), de partager les données entre l'application Java et ces scripts. Par défaut, la **JVM** de **Sun** proposera un interpréteur **JavaScript**, mais d'autre (comme **Python**, **Ruby**, **BeanShell** ou encore **Groovy**) sont d'ores et déjà disponible via le projet scripting : https://scripting.dev.java.net/
Pour plus de détail sur cette JSR, vous pouvez télécharger les spécifications sur la page de la JSR :
http://jcp.org/en/jsr/detail?id=223

JSR 199 : Java Compiler API

Cette API définit un service permettant aux programmes Java d'invoquer un compilateur Java directement dans une application Java, et de récupérer toutes les informations retournées par le compilateur (erreurs, warnings, messages, etc.). Elle ne concerne pas directement le développeur "de base", mais cible principalement les outils de développement comme les EDIs ou encore les moteurs JSP. D'ailleurs les premiers tests de cette API avec **GlassFish** (l'implémentation de référence de **Java EE 5**) ont donné des temps de compilations trois fois plus rapides grâce à l'accès direct au compilateur.
Son API utilise le package **javax.tools** et vous trouverez ses spécifications sur la page de la JSR : http://jcp.org/en/jsr/detail?id=199

JSR 269 : Pluggable Annotation-Processing API

Cette API permet de traiter les annotations lors de la compilation des codes sources, en interaction avec le compilateur. Il ne s'agit pas d'une nouveauté en soit, puisqu'un tel mécanisme était présent dans Java 5.0 avec APT (Annotation Processing Tool). Toutefois, son API et les possibilités offertes ont été étendues et standardisées. Cette API autorisera l'écriture de "librairies intelligentes" qui pourront communiquer avec le compilateur afin de signaler des erreurs d'utilisation ou encore de générer dynamiquement des classes et/ou des fichiers de configuration.
Elle utilise les packages **javax.annotation.processing** (pour le traitement des annotations à l'exécution) et **javax.lang.model.*** (qui représente le code lors de sa compilation).
Pour plus de détail : http://jcp.org/en/jsr/detail?id=269

JSR 250 : Common Annotations

Cette JSR vient définir cinq nouvelles annotations standards, qui pourront également être utilisées par la plateforme entreprise (**Java EE**). Elles serviront par exemple à marquer les codes source générés dynamiquement ou encore à simplifier le déploiement des ressources.
Le détail des spécifications est librement téléchargeable : http://jcp.org/en/jsr/detail?id=250

JSR 202 : Class File Specification Update

Le format des fichiers *.class a été mis à jour afin de bénéficier de l'architecture utilisée dans la plateforme mobile (**Java ME**), qui apporte des avantages inhérents aux performances, aussi bien en terme d'espace utilisé (90%) que du temps de chargement de ces

dernières (approximativement 2 fois plus rapide). Ce schéma est également plus simple et plus robuste, et facilitera les évolutions futures de la plateforme.
Toutes les informations sur ce nouveau format sont librement téléchargeables : http://jcp.org/en/jsr/detail?id=202

JSR 221 : JDBC 4.0 API Specification

Les spécifications de **JDBC 4.0** viennent apporter un vent de fraicheur sur l'API standard d'accès aux bases de données de Java, en utilisant les facilités du langage apportées par Java 5.0, avec notamment un mapping objet/relationnel très simple basé sur les annotations, le support de nouveaux types SQL (**SQL ROWID** et **SQL XML**), l'amélioration du support des **BLOB** (Binary Large Object) et **CLOB** (Character Large Object) ou encore une nouvelle hiérarchie des **SQLExceptions**.
Les spécifications de **JDBC 4.0** sont bien sûr également disponible : http://jcp.org/en/jsr/detail?id=221

Les JSRs concernant XML et Web Services

Comme prévu, **Java SE 6** sera fortement axé vers XML et les Web Services, avec ni plus ni moins que 5 JSRs sur le sujet, brièvement présentées ici :

JSR 105 : XML Digital-Signature API

Cette API, représentée par le package **javax.xml.crypto**, est une implémentation de la norme XML Digital-Signature du W3C (http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/) et est une clef importante de la sécurité des Web Services. De plus elle est requise par d'autres JSR de cette section. Plus d'info : http://jcp.org/en/jsr/detail?id=105

JSR 173 : Streaming API for XML (StAX)

StAX est une API permettant d'exploiter les documents XML tout comme **DOM** ou **SAX** dont elle reprend les principaux avantages de chacun. **StAX** lit le document de manière linéaire afin de ne pas encombrer la mémoire (comme **SAX**), par contre la lecture du document n'est pas évènementielle mais c'est l'application qui détermine les éléments à lire (à l'instar de **DOM**). Plus d'info : http://jcp.org/en/jsr/detail?id=173

JSR 181 : Web-Services Metadata for the Java Platform

Cette JSR définit un ensemble d'annotations permettant de décrire facilement des Web-Services au niveau du code source. Ces annotations détermineront comment les serveurs d'applications déploieront ces Web-Services. Plus d'info : http://jcp.org/en/jsr/detail?id=181

JSR 222 : Java Architecture for XML Binding (JAXB) 2.0

Il s'agit d'une révision majeure des spécifications de **JAXB 1.1**, l'API permettant de générer des classes Java à partir de **XML Schema** et inversement. Cette version met l'accent (entre autres) sur le support de la norme des **XML Schema**. Plus d'info : http://jcp.org/en/jsr/detail?id=221

JSR 224 : Java API for XML-Based Web Services (JAX-WS) 2.0

Cette spécification étend **JAX-RPC 1.0** afin de simplifier le développement, de mieux s'intégrer avec **JAXB 2.0** et d'apporter le support des derniers standards (**SOAP 1.2**, **WSDL 2.0** et **WS-I Basic Profile 1.1**). Plus d'info : http://jcp.org/en/jsr/detail?id=224

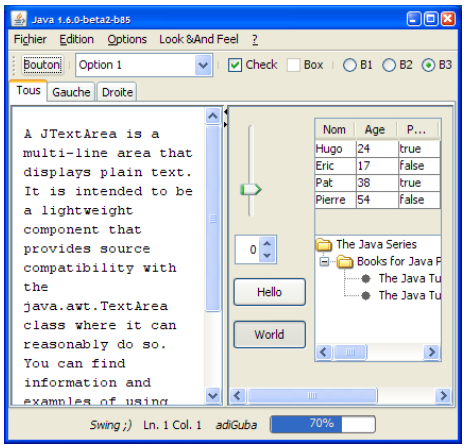
Et les APIs existantes ?

Les APIs existantes ne sont pas en reste, et on notera principalement les changements suivants :

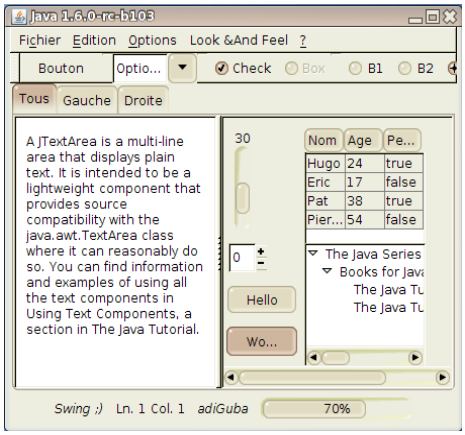
AWT et l'intégration au système

AWT permettra désormais une meilleure intégration au système, en gérant des fonctionnalités optionnelles (c'est à dire qui dépendent du système d'exploitation qui fait tourner la JVM) tels que l'ajout d'icône dans la zone de notification ("System Tray"), le lancement du navigateur ou du client mail par défaut, l'ouverture de fichier selon les mécanismes d'association du système, ou encore un système de modalité des boites de dialogues plus évolué avec plusieurs types de modalité et la possibilité d'exclure des fenêtres (c'est à dire d'empêcher qu'elle soit bloquée par une boite de dialogue modale).

Swing - Les principales nouveautés



Swing s'intègrera également mieux avec l'amélioration de ses **LookAndFeels** systèmes (**Windows** et **GTK**) qui utilise les API système afin de récupérer les composants graphique du thème utilisateur. On note également un support de l'impression amélioré pour les composants textes, la possibilité de trier et filtrer les JTables**,** d'utiliser n'importe quel composant comme onglet sur les **JTabbedPanes**, l'intégration du **GroupLayout** (le **LayoutManager** utilisé par **Matisse**, le GUI-Builder de **NetBeans**) ou encore de la classe **SwingWorker** destiné à faciliter la gestion des tâches.



L'API de Collection

L'API de **Collection** est étendue, notamment avec les interfaces **Deque** (Queue à double sens), **NavigableSet** et **NavigableMap** (respectivement des **SortedSet** et **SortedMap** avec une navigation à double sens et des méthodes de recherches évoluées).

Entrées/Sorties et Réseaux

La classe **File** permet désormais d'obtenir des informations sur l'espace disque occupé et disponible, ainsi que de modifier les attributs des fichiers (read/write/execute).
Au niveau réseau, la classe **NetworkAddress** fournit désormais plus d'informations sur la configuration réseau (adresse de broadcast, masque réseau, adresse MAC, taille du MTU, etc.), les noms de domaine internationalisé sont désormais gérés via la classe **java.net.IDN** et la classe **CookieManager** permet de gérer simplement les **Cookies** lors d'une transaction **HTTP**.

Localisation et internationalisation

Au niveau de la localisation, on note l'ajout de nouvelles locales et du calendrier impérial chinois, mais on retiendra surtout la possibilité de contrôler le cache et le comportement des **ResourceBundles** ou la nouvelle classe **java.text.Normalizer** qui permettra de transformer les chaînes Unicode de leurs formes composées vers leurs formes décomposées et vice-versa.

JVMTI (JVM Tool Interface)

JVMTI (JVM Tool Interface) améliore encore la surveillance des applications Java, en offrant la possibilité d'accéder au contenu du heap et en autorisant désormais l'**Attach-on-Demand** afin de surveiller n'importe quelle application Java sans la redémarrer.

Divers

La nouvelle classe **java.util.ServiceLoader** permet de gérer un système de service tel qu'il est déjà utilisé par plusieurs classes de l'API standard, en recherchant toutes les implémentations concrètes d'une interface (ou d'une classe abstraite) déclarée dans un des fichiers des répertoires **META-INF/services** de chacun des éléments du **CLASSPATH**. Ceci est utilisé par exemple pour rechercher automatiquement les drivers **JDBC**.
Calcul à virgule flottante : ajout de méthodes recommandées par l'**IEEE 754**. Ainsi les classes **Math** et **StrictMath** se sont vue rajoutées les méthodes **scalb()**, **getExponent()**, **nextAfter()**, **nextUp()**, et **copySign()** pour les types **float** et **double**.
Le package java.util.concurrent introduit dans Java 5.0 a reçu quelques correctifs mineurs, comme l'ajout de nouvelles unités de temps dans la classe **TimeUnit** : **MINUTES**, **HOURS** et **DAYS**.

On retiendra également le support des images au format **GIF** en écriture (jusqu'à présent seul le support en lecture était disponible en standard) qui est devenu possible avec l'expiration des brevets le concernant (le format est dorénavant dans le domaine public).
Enfin la définition du **CLASSPATH** supporte désormais le wildcard * qui représente toutes les archives jar/zip d'un répertoire. Ainsi si le **CLASSPATH** (via la ligne de commande, la variable d'environnement ou l'attribut du manifest du Jar) comporte par exemple **lib/*** toutes les archives du répertoire lib seront ajoutées au Classpath de l'application, et il sera donc inutile de les nommer une à une.

Pour aller plus loin...

Si le sujet vous intéresse et que vous voulez approfondir les choses, je ne peux que vous conseillez d'aller jeter un coup d'oeil aux différentes JSRs ou sur le site officiel :

- **Java™ SE 6 Release Notes - Features and Enhancements** : http://java.sun.com/javase/6/webnotes/features.html
- **JDK 6 Documentation** : http://java.sun.com/javase/6/docs/
- **Java Platform API Specification** : http://java.sun.com/javase/6/docs/api/

Et pourquoi pas de venir en parler avec nous sur le forum **Java** de **Developpez.com** : http://www.developpez.net/forums/forumdisplay.php?f=22

Lire l'article d'adiguba en ligne : Lien15

NetBeans 5.5 disponible, NetBeans 6.0 déjà en chantier

NetBeans 5.5 est sorti fin octobre de cette année. Un peu après IntelliJ Idea 6.

A cette occasion, le site de NetBeans a été fortement revu, essayant de simplifier fortement le premier contact avec leurs nouveaux utilisateurs.

NetBeans a toujours mis l'accent sur l'effet "out-of-the-box". Pour l'équipe de NetBeans, il est important que l'utilisateur puisse travailler avec l'EDI NetBeans dès son installation terminée.
Voyons donc ce que NetBeans 5.5 nous offre une fois son installation terminée:

- Un éditeur visuel de GUI (nom de code Matisse)
- Gestion de projet basée sur Ant
- Support pour CVS
- Débogueur
- Édition des sources Java, JSP, HTML, XML, ...
- Refactoring de code poussé
- Développement d'applications Web en utilisant JSP, JSF et Struts
- Développement des EJB (NetBeans 5.5 met l'accent sur les EJB 3.0)
- Développement de Web Services
- Développement de modules ou d'applications RCP
- Des exemples d'applications,
- le catalogue Java Blue Print

Mais NetBeans 5.5. ce n'est pas que cela, c'est également des Packs dédiés.

- le Mobility Pack. Ou plutôt, je devrais dire les Mobility Pack, car vous en avez un dédié pour les plateformes CLDC et un autre pour les plateformes CDC.
- le Visual Web Pack, qui vous permet de développer des applications JSF en utilisant le moteur visuel qui a fait le succès de Sun Java Studio Creator.
- l'Entreprise Pack, qui vous donnes des outils pour le développement SOA principalement (support Web Services, WSDL, XML Schema, BPEL, ...)
- Le profiler pack, qui vous permet de profiler vos applications, et découvrir rapidement où se situent vos problèmes de performances ou vos memory leaks.
- Le C/C++ pack, qui vous permet de créer des applications C/C++ depuis l'EDI NetBeans

Mais c'est également un Centre de mise à jour qui vous permet d'installer d'autres modules, comme par exemple

- Le support de SubVersion
- le module Jackpot que je nommerais le refactoring de nouvelle génération.
- Un module de collaboration en ligne
- la modélisation UML
- un éditeur visuel encore plus aboutit

Critique de Vincent Brabant

Il est encore plus facile, maintenant que le site de netbeans a été revu, de télécharger l'EDI NetBeans et l'installer. Installation qui s'est déroulée sans problèmes. Une fois l'installation terminée, on a droit à un écran de bienvenue qui, si vous êtes connecté à Internet, se mettra à jour automatiquement, en vous affichant les dernières nouvelles et articles provenant du site netbeans.org, mais aussi en voyant les billets des différents bloggueurs recensés sur planetnetbeans. Ce qui vous permet de suivre l'actualité de

NetBeans depuis votre EDI même. Très pratique.

Pour le reste, j'aurais envie de dire que rien n'a vraiment changé par rapport à NetBeans 5.0, de premier abord. Car en fait, NetBeans 5.5 rajoute le support de Java EE 5, ce que NetBeans 5.0 n'avait pas.

L'assistant pour les nouveaux projets s'en fortement enrichit au niveau des exemples. Ainsi, dans les solutions blueprint, vous avez droit à des exemples concernant Ajax, mais aussi concernant JPA. Un nouvel exemple de module NetBeans nous est également fournit. Ce qui est vraiment super pour l'étude des API NetBeans.

J'ai également remarqué que NetBeans 5.5 à l'air de mieux supporter Java SE 5.0 que IntelliJ par exemple. NetBeans 5,5 me permet par exemple de créer la classe package-info.java, alors que je n'ai jamais réussi à le faire avec IntelliJ.

Une autre nouveauté que j'ai particulièrement appréciée, est le fait de pouvoir créer son modèle de projet. Ainsi, vous pouvez créer votre modèle de projet et le partager ensuite avec le restant de l'équipe.
Point de vue éditeur, je n'ai pas vraiment constaté d'améliorations par rapport à NetBeans 5.0. Et je n'ai noté aucun changement au niveau de Matisse (l'éditeur visuel des GUI).
Vous pouvez bien évidemment mettre à jour Matisse pour profiter de sa version 1.4, mais cela était également possible sous NetBeans 5.0.
Par contre, le support CVS de NetBeans 5.5 intègre par défaut les corrections qui étaient proposées en mise à jour pour NetBeans 5.0.
Le support UML qui est maintenant disponible sans devoir installer le pack Entreprise est vraiment super. Sa facilité d'utilisation m'a rappelé TogetherJ, ce qui restait pour moi jusqu'à présent la référence pour la facilité d'utilisation. J'arrive maintenant avec le support UML de NetBeans d'avoir le Sequence Diagram de mon bout de code aussi facilement que lorsque j'ai utilisé TogetherJ pour la toute première fois. Ce qui démontre bien sa facilité d'utilisation.
L'autre grande nouveauté de NetBeans 5.5, c'est le support de Subversion. Mais je n'ai malheureusement pas encore eu l'occasion de le tester.

Conclusion

NetBeans 5.5 est une mise à jour mineure comparé à NetBeans 5,0 si l'on regarde au niveau de projets Java SE et Java ME. Mais pour ce qui est des projets Java EE, c'est une mise à jour majeure, avec le support de Java EE 5.0 (EJB 3.0, JPA, WebServices, ...) out of the box, mais surtout le support SOA via l'Entreprise Pack, et l'édition visuelle des pages web via le Visual Web Pack.

L'avenir

L'avenir de NetBeans est déjà en train de se dessiner avec la milestone M5 de NetBeans 6.0 J'ai eu l'occasion de "jouer" avec le nouvel éditeur Java de NetBeans 6.0, et je peux vous assurez que plus personne ne pourra dire qu'il n'utilise pas NetBeans car son éditeur n'est pas aussi bon que celui d'Eclipse ou IntelliJ. J'ose même affirmer que l'éditeur de NetBeans a pris une certaine avance sur la concurrence. Mais plutôt que de croire en mes affirmations, je vous propose de le découvrir par vous-même avec la Milestone 5 de NetBeans 6.

Lire la critique de Vincent Brabant en ligne : Lien16

JetBrain IntelliJ Idea 6

Au mois d'octobre JetBrain annonçait la sortie de la version 6 d'IntelliJ Idea. Parmis les nouveautés, il y a le support de Java EE 5.0. Deux de nos rédacteurs ont déjà eu l'occasion d'en faire le tour: L'un avec un passé Eclipse. L'autre avec un passé NetBeans. Je vous laisse découvrir ci-dessous des extraits de leur critique, dont l'intégralité est disponible en ligne.

Critique de Baptiste Wicht

Tout d'abord, qu'est ce qu'IntelliJ Idea ? C'est un EDI très puissant pour Java, la Rolls des EDI en quelque sorte. Son seul défaut est d'être payant, mais ce défaut est largement compensé par ses énormes possibilités. Je trouve personnellement qu'il vaut tout à fait son prix.
Les principales nouvelles fonctionnalités de la version 6.0 sont un éditeur de Gui très puissant qui supporte beaucoup de layouts (dont le FormLayout), le support des EJB 3.0, un module pour faciliter la gestion en groupe et le calcul de la couverture du code par les tests.
Je vais donc vous présenter toutes ces nouvelles fonctionnalités. Et vous faire découvrir les avantages et inconvénients de cet EDI.

Productivité et fonctionnalités d'édition

Dans Idea, tout a été fait pour permettre d'augmenter la productivité des développeurs. Ainsi, la complétion est déjà bien remplie au départ; vous pouvez par exemple employer psvm pour créer une nouvelle méthode main ou encore sout pour System.out.println(), vous pouvez aussi créer vos propres complétions. La complétion des noms de classes et des variables et aussi pratique.
Mais une des fonctionnalités très intéressantes est la génération de code. En effet, via le raccourci Alt+Insert, vous pouvez directement ajouter un constructeur, des méthodes getters/setters, surcharger une méthode de la superclasse ou encore générer les méthodes equals et hashcode. Cela va générer du code automatiquement, par exemple, pour le constructeur, vous aurez le choix entre tous les constructeurs de la superclasse et vous pouvez directement demander au programme d'initialiser vos variables locales dans le constructeur et de lui ajouter un paramètre. C'est pareil pour les méthodes equals et hashcode, elles vont être directement générées et complétées à partir de vos variables.

Points noirs

Tout d'abord, au début, ce programme n'est pas extrêmement facile à prendre en main. Il faudra un moment avant que vous vous y habituiez et que vous bénéficiez de tous les avantages au niveau de la productivité que permet Idea.
Ensuite, ce programme est plutôt conséquent et consomme beaucoup de mémoire. Il n'est donc pas fait pour des machines ayant une faible puissance.
On peut aussi regretter le fait qu'il n'y ait pas de version en français, mais tout bon développeur maîtrisant les bases du langage de Shakespeare, il ne devrait pas y avoir trop de problèmes pour maîtriser cet Edi qui se révèle très simple d'utilisation dès que vous avez commencé à le maîtriser et très complet.
Enfin, le prix reste tout de même assez important pour un privé, n'utilisant un Edi que pour ses propres développements. Mais, il faut ce qu'il faut, la qualité à un coût.

Conclusion

En conclusion, ce logiciel vous permettra de gagner beaucoup de temps et d'être plus productif. Bien qu'un peu compliqué au début, il vous séduira bientôt par ses multiples possibilités et son aide à la productivité.
Pour résumé cet article, je vais vous présenter en quelques phrases ses avantages et inconvénients.

Avantages

- Un GUI Designer extrêmement puissant
- Une interface très bien pensée et axée rapidité d'accès aux diverses fonctions
- Beaucoup de fonctionnalités facilitant le travail en groupe
- Un très bon support de Java EE
- Des possibilités de refactoring très puissantes

Inconvénients

- Le prix est tout de même assez conséquent pour une utilisation personnelle
- Par de version en français
- Pas très intuitif au début

Lire la critique de Baptiste Wicht en ligne : Lien17

Critique de Valere Dejardin

Mon EDI de choix est Netbeans, que j'utilise en version 5.0 et 5.5, donc ma revue sera axée sur les différences entre les deux outils. Je ne connais vraiment pas assez Eclipse pour pouvoir extrapoler sur la présence de telle ou telle fonctionnalité. Par ailleurs, je me limiterais aux parties de Java que j'utilise au jour le jour, c'est à dire du POJO et du JSP/Servlet.

Mon environnement de travail est Win XP SP2, avec un processeur Pentium D 2.80GHz et 2 Go de RAM.

Déceptions...

Retournons maintenant à l'endroit où on passe le plus de temps: l'éditeur. C'était ,d'après tout ce que j'avais entendu et lu, le point fort de IDEA. Et bien j'ai été assez déçu. Par exemple, les templates par défaut des nouveaux fichiers sont pauvres. Pas de template de classe exécutable avec main() intégré, pas de constructeur précréé dans les nouveaux fichiers. Bien entendu, tout cela est paramètrable, mais c'est dommage. Le debugger est fonctionnel, mais il manque un bouton "apply code change" (pour réinjecter une correction de code en cours de débug sans redémarrer l'application) que j'utilise énormément dans Netbeans. Autres petite irritations, parmi la kyrielle de raccourcis clavier pour naviguer dans les fichiers, ALT+TAB ne fonctionne pas, il faut utiliser ALT+Droite/Gauche; appuyer sur la touche ";" ne place pas ledit ";" en fin de ligne...

Par ailleurs, IDEA a tranché de manière originale en ce qui concerne la polémique "Où placer le bouton pour fermer un onglet?" (cf. le changement du design entre les onglets de Firefox 1 et Firefox 2): plutôt que d'hésiter entre placer le bouton dans chaque onglet ou le placer tout à droite, ici il n'y a carrément pas de bouton du tout... le clic droit est obligatoire. Peut-être une manière de forcer les utilisateurs a abandonner la souris pour le clavier?

Plus gênant, j'ai vraiment eu du mal avec le code completion: déjà, il a fallu trouver l'option pour afficher la javadoc en même temps que le code completion. Ensuite, lorsque je souhaite saisir les arguments d'une méthode, pas moyen d'afficher la javadoc ou même la signature de cette méthode! Par exemple, dans Netbeans, si je souhaite créer un nouveau PreparedStatement, la javadoc me propose la liste des différents arguments possibles, alors que IDEA me propose une sélection d'objets dans la portée de mon code:

Certes le code completion est rapide, mais il n'affiche pas ce que je souhaite!

Agréables surprises

Malgré tout, certaines idées sont excellentes et bien pratiques. Par exemple, l'indentation automatique lors du collage d'un bout de code n'a l'air de rien, mais permet d'économiser le CTRL-SHIFT-F qui est devenu un réflexe pour moi dans Netbeans. Une autre fonctionnalité intéressante, mais que je n'ai pas eu le réflexe d'utiliser: l'option Local History qui permet de suivre l'évolution locale d'un fichier, sans passer par CVS ou SVN. Si on reste dans l'intégration CVS, la fenêtre Diff est très bien faite.
Dans la catégorie "IDEA automatise les tâches que je faisais à la main", l'assistant 'commit' intelligemment d'exécuter avant le commit des actions comme l'optimisation des imports, un reformatage du code, une analyse du code... Sachant que dans Netbeans je réalisais moi même ces actions (avec PMD pour l'analyse du code) je ne peux qu'applaudir.
Une autre délicate attention de IDEA envers ses utilisateurs est l'affichage à l'écran, lors des actions longues (commit, lancement Tomcat, scan du classpath) d'un écran de trucs et astuces qui permet de se familiariser avec les possibilités de l'outil.

En conclusion

Avec la sortie de Netbeans 5.5 et de IDEA, la mode est aux comparatifs entre les différents EDI, mais à la différence des participants de l'expérience "Essayez l'autre EDI pendant 30 jours" (dont je n'avais pas connaissance au début de ma propre expérience), je n'ai pas consacré autant de temps à ma revue. J'avoue être repassé sous Netbeans lorsque je n'arrivais pas à trouver quelque chose rapidement.

Je suis certain que certains des problèmes que j'ai évoqué dans cette revue peuvent être résolus pas les aficionados de IDEA, de la même manière que j'ai tiqué parfois en lisant les commentaires des deux utilisateurs Eclipse qui testaient Netbeans. J'ai voulu faire une revue et en particulier sur la partie de IDEA qui est censée être loin devant Netbeans: l'éditeur. Et la surprise a été que je n'ai pas été spécialement impressionné. Certes, au niveau des détections d'erreurs, des inspections de code, IDEA conserve de l'avance, mais pas tant que cela, mais j'avais parfois l'impression de n'y voir que du saupoudrage de sucre.... Il faut croire que le si décrié éditeur de Netbeans ait quand même fait bien des progrès depuis l'antique Netbeans 3.5.1.

Au final, IDEA est un EDI de grande qualité: interface claire est nette, fonctionnalités embarquées assez similaires au Netbeans de base (J2SE, J2EE, Web, support de Struts et JSF, éditeur de GUI - que je n'ai pas testé), éditeur java et JSP de grande qualité. Je vais pour ma part continuer à travailler avec Netbeans - j'y ai mes aises - mais pour celui qui sait bien l'utiliser, IDEA présente à mes yeux une alternative tout à fait crédible.

Au final, comme souvent dans ce genre d'exercice, le choix d'un EDI reste quelque chose de personnel, une affaire de goût.

Lire la critique de Valere Dejardin en ligne : Lien18

Le garbage collector

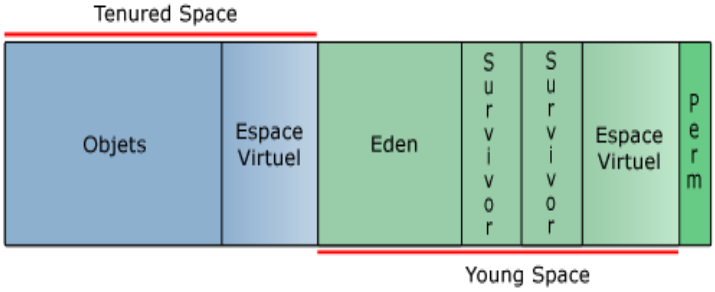
Le Garbage Collector, que je nommerai par la suite GC pour ne fatiguer ni vos yeux ni mes doigts, est une sentinelle bienveillante présente dans toutes les machines virtuelles. Son rôle consiste à identifier puis à libérer les zones de mémoires inutilisées par l'application en cours d'exemple. Ceci explique en partie son affreux nom de ramasse-miettes.

Au cours de son cycle de vie, une application crée une certaine quantité d'objets, c'est-à-dire des données qui consomment de la mémoire, dont la durée de vie varie suivant leur rôle au sein du programme. Par exemple, l'objet correspondant à la fenêtre de votre navigateur Internet possède, en simplifiant, une durée de vie équivalente à celle de l'application. A l'inverse l'objet représentant le logo #ProgX sur cette page ne vit que le temps que vous passez sur ce site. De manière générale une application crée un grand nombre d'objets à courte durée de vie. Quoi qu'il en soit, vous comprendrez aisément que déterminer et contrôler le cycle de vie de chaque objet dans un programme demande un effort considérable de la part du programmeur.

Pour vous donner une idée de l'incroyable quantité d'objets mis au monde puis assassinés, sachez que la simple lecture d'un fichier dans un éditeur de texte en nécessite 342 997. La simple conversion d'un quintal en kilogrammes dans le logiciel NumericalChameleon entraîne la vie et la mort de 171 896 objets. Même si le programmeur ne doit pas explicitement manipuler dans sa tête autant d'objets une simple erreur de sa part peut se révéler catastrophique. Telle est la cause des fameuses fuites mémoire dont souffrent certains programmes. Celles-ci arrivent quand un grand nombre d'objets créés ne sont jamais détruits : le programme prend de plus en plus de mémoire jusqu'à effondrement. Une bonne gestion de la mémoire est l'une des plus grosses difficultés relatives à des langages comme le C ou le C++. Java, comme SmallTalk ou Python, repose sur un GC dont le rôle est de diviser la quantité de travail du programmeur par deux. Seule la création des objets l'intéressera alors.

Bien qu'extrêmement pratique, un GC n'a rien de magique et entraîne souvent des effets pervers. A l'abri derrière ce bouclier contre la gestion de la mémoire, le développeur peut facilement oublier ses bonnes manières et obtenir des résultats catastrophiques. Il pourra alors blâmer le GC, le langage ou encore la plate-forme. En tâchant de comprendre comment fonctionne le GC vous pourrez dans un premier temps optimiser la machine virtuelle Java (JVM) pour l'adapter aux spécificités de chacune de vos applications. Comme je vous l'ai dit nous ne verrons pas de code ici mais, si le sujet vous intéresse toujours, nous pourrons nous tourner vers des détails plus techniques dans une prochaine news.

Les GC existent depuis très très longtemps en informatique et des dizaines d'algorithmes existent. Puisque nous aimons tous être à la page, je ne traiterai ici que du cas des JVM HotSpot 1.4.1 et supérieures de Sun Microsystems. Vous devez comprendre que chaque auteur de JVM peut choisir sa propre stratégie pour le GC. En outre, pour une même version d'une même JVM, certains détails peuvent varier d'une application à l'autre. Ainsi les programmes Java exécutés sous Solaris pourront gagner en performances en s'intéressant de près au problème des *threads*. Le GC de la machine virtuelle HotSpot qui nous intéresse est appelé *generational garbage collector*. Il s'agit, en français, d'un ramasse-miettes faisant la différence entre plusieurs générations d'objets. Dans la machine virtuelle, les objets naissent, vivent et meurent dans une zone mémoire appelée *heap*, ou tas. Ainsi le *heap* se compose de deux partie correspondant au deux générations d'objets possibles : le *young space* et le *tenured* (ou *old*) *space*. Le premier accueille les objets récents, ou enfant, tandis que le second contient les objets à longue durée de vie, les ancêtres. Outre le *heap*, la JVM exploite une autre zone mémoire appelé *perm* dans laquelle est archivé le code binaire de chaque classe chargée pour l'exécution du programme. Bien que le *perm* soit très important pour les applications réalisant beaucoup de génération dynamique de code, comme les serveurs JSP, nous ne nous y attarderons pas plus longtemps. Consultez plutôt le schéma ci-dessous résumant la structure du *heap*.



Les espaces virtuels correspondent à de la mémoire disponible pour la JVM mais non occupée par les données. La taille du *young* et du *tenured space* peut en effet varier dans le temps. Cette caractéristique a une importance capitale sur le paramétrage de la JVM. Sur ce schéma se trouvent trois zones supplémentaires, incluses dans le *young space* : l'éden et deux *survivor spaces*. Pour comprendre leur utilité nous allons devoir nous intéresser à présent aux algorithmes utilisés par le GC.

Quand un nouvel objet est alloué sur le tas, la JVM le crée dans l'éden. Les *survivor* sont pour leur part occupés à tour de rôle. Celui resté libre sert de réceptacle pour favoriser le travail du GC. Lorsque le taux d'occupation du *young space* devient inquiétant, le GC exécute une *minor collection*. Pour ce faire, un algorithme de copie, très simple, est utilisé et c'est ici qu'intervient le *survivor* libre. Le GC va en effet parcourir tous les objets de l'éden et du *survivor* occupé pour déterminer quels objets sont encore en vie, c'est-à-dire quels objets possèdent encore des références externes. Chacun d'entre eux sera alors copié dans le *survivor* resté vide. L'avantage majeur de cet algorithme réside dans sa vitesse d'exécution car il n'y a pas de libération de la mémoire à proprement parler. Après une *minor collection*, l'éden et un *survivor space* sont considérés comme libres. Le travail de copie est pour sa part favorisé par une caractéristique des JVM actuelles qui implique que l'ensemble du *heap* ne forme qu'un seul segment continu de mémoire. Ceci explique notamment pourquoi, sous

Windows, le *heap* ne peut dépasser 1,5 Go environ. Au fil des *minor collections*, les objets restés en vie passent d'un *survivor space* à l'autre. Lorsqu'un objet atteint un âge suffisant, déterminé dynamiquement par *HotSpot* à l'exécution, ou lorsque le *survivor space* d'accueil n'est plus assez grand, une copie est effectuée vers le *tenured space*. La plupart des objets naissent et meurent directement dans le *young space*. Pour obtenir des performances idéales, il est préférable de configurer la JVM pour éviter des copies inutiles du *young space* au *tenured*.

Dans ce nouvel espace mémoire les lois ne sont plus les mêmes. Lorsque de la mémoire est requise, une major collection est effectuée à l'aide de l'algorithme *Mark-Sweep-Compact*. Ce dernier n'est pas très compliqué mais bien plus gourmand que celui de copie. En résumé, le GC va parcourir tous les objets du *heap*, marquer les candidats à l'extermination, et parcourir tout le *heap* de nouveau pour tasser les objets encore en vie et ainsi éviter la fragmentation de la mémoire. Parcourir le *heap* par deux fois est la cause principale des baisses de performances que l'on peut parfois percevoir dans des applications Java. Pour effectuer son travail le GC doit en effet interrompre complètement l'exécution de l'application. Vous comprendrez sans peine qu'il est judicieux de réduire la charge du *Mark-Sweep-Compact* au maximum.

Compte tenu de ces informations, vous pouvez maintenant utiliser les options en ligne de commande suivantes pour la JVM, où taille peut être une taille en octets (32765), en kilo-octets (96k), en méga-octets (32m) ou encore en giga-octets (2g) :

- -Xms[taille], définit la taille minimale du *heap*. Ce paramètre permet d'éviter des dimensionnements fréquents du *heap* si vous savez que votre application utilise beaucoup de mémoire,
- -Xmx[taille], définit la taille maximum du *heap*. Ce paramètre est majoritairement utilisé pour les applications serveurs qui nécessitent parfois plusieurs giga-octets de mémoire. Le *heap* peut varier entre les valeurs spécifiées par -Xms et -Xmx,
- -XX:NewRatio=[nombre], indique le rapport de taille *tenured* sur *young space*. Le nombre 2 donnera par exemple un *tenured* de 64 Mo et un *young* de 32 Mo pour un *heap* global de 96 Mo,
- -XX:SurvivorRatio=[nombre], indique le rapport de taille entre l'éden et un *survivor space*. Pour un ratio de 2 et un *young space* de 64 Mo, l'éden occupera 32 Mo et chaque *survivor* 16 Mo.

Le choix des valeurs pour chaque paramètre dépend énormément de votre application. Une application créant une grande quantité d'objets à courte durée de vie, un serveur HTTP par exemple, n'aura pas les mêmes besoins qu'une application très statique, comme un économiseur d'écran. Référez vous au document officiel de Sun pour obtenir une liste complète des paramètres de la JVM HotSpot.

Outre le contrôle du *heap*, HotSpot permet de modifier le comportement même du GC. Bien que les algorithmes ne changent pas vraiment, il existe trois modes particuliers d'exploitation du GC. Nous savons qu'en temps normal l'exécution de l'application est interrompue pour effectuer les *minor* et *major collections*. Ce comportement est très pénible sur des machines multi-processeurs par exemple. Le premier mode est le GC incrémental. Dans ce cas le GC effectue une partie d'une *major collection* à chaque *minor collection*. Les performances globales sont réduites mais les longues attentes lors des major collections sont éliminées. Comme ce mode peut entraîner une grande fragmentation du *heap* dans le *tenured space*, il est conseillé de l'utiliser dans des applications possédant surtout des objets à longue durée de vie. Le deuxième mode correspond au GC parallèle qui utilise plusieurs *threads*, un par processeur par défaut, pour les *minor collections*. Ce mode ne trouve pas d'intérêt en dessous de 3 processeurs. Le troisième et dernier mode est le GC concurrent. Il permet d'effectuer les major collections de manière incrémentale mais sans, pour ainsi dire, interrompre l'application. Le GC concurrent peut également bénéficier du traitement parallèle des *minor collections*. D'après les documents de Sun Microsystems ce GC obtient de bons résultats avec des applications interactives sur architecture mono processeur. Voici les options permettant d'utiliser ces différents modes :

- -Xincgc, active le GC incrémental,
- -XX:+UseParallelGC, active le GC parallèle. Le nombre de *threads* utilisé peut être modifié à l'aide de l'option -XX :ParallelGCThreads=[nombre],
- -XX:+UseConcMarkSweepGC, active le GC concurrent. Les *minor collections* parallèles peuvent être activés en utilisant conjointement l'option -XX:+UseParNewGC.

Pour mesure l'impact de vos paramètres vous pouvez ajouter l'option -verbose:gc à la ligne de commande exécutant votre application. Vous obtiendrez une trace de l'activité du GC sur la sortie standard.

Lire l'article de Romain Guy en ligne : Lien19

La sérialisation XML facile avec l'API XStream

Les développeurs Java sont souvent amené à sérialiser leurs objets. L'API XStream est là pour vous faciliter la sérialisation XML.

Introduction

XStream est une API Java qui permet de sérialiser et désérialiser des objets dans des fichiers XML. Les avantages d'XStream sont principalement :

- La facilité d'utilisation de l'API.
- Pas de modification de code des objets que vous voulez sérialiser.
- La rapidité d'exécution et une faible utilisation de la mémoire.

Toutes les sources de cet article sont disponibles ici[Lien20] sous la forme d'un projet pour Eclipse[Lien21].

Où trouver XStream ?

XStream est disponible à l'adresse suivante http://xstream.codehaus.org.
XStream est téléchargeable sous la forme d'une librairie JAR par le menu "Download" du site.
Pour la suite de l'article, nous utiliserons la dernière version stable **xstream-x.x.jar** obtenue sur le site (Télécharger ici[Lien22]).
La librairie **xstream-x.x.jar** est à mettre dans le classpath pour que les exemples fonctionnent.

Création des classes à sérialiser

Nous allons premièrement créer deux classes, **Entete** et **Article**.
La classe **Article** déclarera un attribut de classe de type **Entete**.
Ces deux classes contiennent les constructeurs, getters et setters utiles.
La classe **Entete** :

```
package beans;
import java.util.Date;
public class Entete {
    private String titre;
    private Date dateCreation;
    public Date getDateCreation() {
        return dateCreation;
    }
    public void setDateCreation(Date dateCreation) {
        this.dateCreation = dateCreation;
        }
    public String getTitre() {
        return titre;
    }
    public void setTitre(String titre) {
        this.titre = titre;
    }
    public Entete(String titre, Date dateCreation) {
        super();
        this.titre = titre;
        this.dateCreation = dateCreation;
    }
}
```

Et la classe **Article** qui utilise la classe **Entete** :

```
package beans;
public class Article {
    private Entete entete;
    private String synopsis;
    public String getSynopsis() {
        return synopsis;
    }
    public Article(Entete entete, String synopsis) {
        super();
        this.entete = entete;
        this.synopsis = synopsis;
    }
    public void setSynopsis(String synopsis) {
        this.synopsis = synopsis;
    }
    public Entete getEntete() {
        return entete;
    }
    public void setEntete(Entete entete) {
        this.entete = entete;
    }
}
```

Sérialisation des classes

Ci-dessous une classe **Serialisation** avec une méthode *main* contenant le code pour convertir les classes précédentes en chaîne XML :

```
package tests;
import java.util.Date;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;
public class Serialisation {
    public static void main(String[] args) {
      // Instanciation de la classe XStream
      XStream xstream = new XStream(new DomDriver());
      // Instanciation de la classe Entete
      Entete entete =
        new Entete("Titre de l'article",new Date());
      // Instanciation de la classe Article
      Article article = new Article(entete,
        "Un synopsis bien placé !!! " +
        "<strong>avec une balise HTML</strong>");
      // Convertion du contenu de l'objet article en XML
      String xml = xstream.toXML(article);
      // Affichage de la conversion XML
      System.out.println(xml);
    }
```

La console affichera :

```
<beans.Article>
  <entete>
    <titre>Titre de l&apos;article</titre>
    <dateCreation>2006-10-01 18:00:31.187
CEST</dateCreation>
  </entete>
  <synopsis>Un synopsis bien placé !!!
&lt;strong&gt;avec une balise
HTML&lt;/strong&gt;</synopsis>
</beans.Article>
```

XStream convertit le contenu de l'objet **article** en un joli fichier XML, on remarquera que les caractères spéciaux sont également traités :

- ' en '
- en <strong>

Maintenant, pour sérialiser l'objet **article** en un fichier *article.xml*, le code de la classe serait :

```
package tests;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Date;
import beans.Article;
import beans.Entete;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;
public class Serialisation {
  public static void main(String[] args) {
    try {
      // Instanciation de la classe XStream
      XStream xstream =
        new XStream(new DomDriver());
      // Instanciation de la classe Entete
      Entete entete =
        new Entete("Titre de l'article",new Date());
      // Instanciation de la classe Article
      Article article =
        new Article(entete,
            "Un synopsis bien placé !!! " +
            "<strong>avec une balise HTML</strong>");
      // Instanciation d'un fichier c:/temp/article.xml
      File fichier = new File("c:/temp/article.xml");
      // Instanciation d'un flux de sortie fichier vers
      // c:/temp/article.xml
      FileOutputStream fos =
        new FileOutputStream(fichier);
      try {
        // Sérialisation de l'objet article dans
        // c:/temp/article.xml
        xstream.toXML(article, fos);
```

```
      } finally {
        // On s'assure de fermer le flux
        // quoi qu'il arrive
        fos.close();
      }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
```

Désérialisation des classes

Maintenant pour recharger (désérialiser) un objet de type Article avec les données sérialisées dans le chapitre précédent, le code sera :

```
package tests;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import beans.Article;
import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;
public class Deserialisation {
  public static void main(String[] args) {
    try {
      // Instanciation de la classe XStream
      XStream xstream = new XStream(new DomDriver());
      // Redirection du fichier c:/temp/article.xml
      // vers un flux d'entrée fichier
      FileInputStream fis = new FileInputStream(
        new File("c:/temp/article.xml"));
      try {
        // Désérialisation du fichier
        // c:/temp/article.xml vers un nouvel
        // objet article
        Article nouvelArticle =
          (Article) xstream.fromXML(fis);
        // Affichage sur la console du contenu de
        // l'attribut synopsis
        System.out.println(nouvelArticle.getSynopsis());
      } finally {
        // On s'assure de fermer le flux
        // quoi qu'il arrive
        fis.close();
      }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
  }
}
```

Lire l'article d'Eric Reboisson en ligne : Lien23

Les blogs Java

Quelques billets trouvés sur les blogs Java

Comprendre les références en Java

L'API Java est vraiment énorme. Elle offre un grand nombre d'outils et de possibilités qui, bien souvent, ne sont pas utilisées tout simplement parce qu'elles sont méconnues...
Cela faisait quelques temps que je voulais m'intéresser plus particulièrement aux diverses références de **Java**, et le récent article sur les références faibles sous .NET[Lien24] m'avait donné l'eau à la bouche... Et voilà qu'un article en anglais les décrit avec des exemples concrets : Understanding Weak References[Lien25]
Je me suis donc jeter dedans et je vais essayer de vous présenter cela.

1. Références fortes (*Strong references*)

En tout bien tout honneur, je commencerais par les *références fortes*, qui ne sont ni plus ni moins que les références ordinaires des objets **Java**, que vous utilisez tous les jours :

```
StringBuffer buffer = new StringBuffer();
```

Le code ci-dessus crée un objet de type **StringBuffer** et stocke une *référence forte* dans la variable *buffer*.
Tant qu'un objet est accessible via une *référence forte* (on dit alors qu'il est "*strongly reachable*"), il n'est pas éligible à la destruction par le Garbage Collector : ce dernier ne supprime pas les objets que vous utilisez, mais seulement ceux que vous n'utilisez plus, c'est à dire lorsque vous n'avez plus de références sur ces objets ou que vous les passez explicitement à **null**...
Toutefois, si ce mécanisme de référence permet de simplifier la gestion de la mémoire, en libérant le developpeur de gérer lui-même la libération des données, cela peut avoir certains désavantages dans certain cas.
Ainsi, avec Java 1.2 est apparut le package **java.lang.ref[Lien26]**, qui définit un ensemble de référence supplémentaire, qui permettent de définir des contraintes moins fortes qui s'avère utile dans certains cas.

Présentation du package java.lang.ref

Reference

Le package **java.lang.ref** défini trois nouveaux types de référence, qui héritent toutes de la classe abstraite **Reference[Lien27]**, qui définit l'ensemble des méthodes communes aux références.
Elle définit en particulier une méthode **get()** qui permet de récupérer une *référence forte* vers l'instance (si elle est disponible).
Ces références donnent moins de contraintes au GC, qui peut donc libérer les objets même s'il sont toujours référencés (sous certaines conditions). Ainsi, lorsque la référence n'est plus valide, sa méthode **get()** renvoi **null**.

ReferenceQueue

Il est possible d'être informé de la non-validité d'une référence en enregistrant cette dernière dans une **ReferenceQueue[Lien28]**. En effet dès que le GC modifie leurs états, les références s'ajoutent à la **ReferenceQueue**, qui propose trois méthodes pour les récupérer :

- La méthode **poll()** renvoit la première référence et la supprime de la queue, ou **null** si la queue ne comporte aucune référence.
- La méthode **remove()** supprime la première référence de la queue, en bloquant le thread courant si elle ne contient aucune référence.
- La méthode **remove(long)** permet exactement la même chose en utilisant un timeout.

Cette classe permet ainsi d'être tenu informée de l'état des références, et permet en particulier de les libérés de leurs *références fortes* (en effet, puisque ces références sont représentées par un objet, elles sont elles-même représentées par un *référence forte*).

Références douces (*Soft references*)

Les *références douces* (représentées par la classe **SoftReference[Lien29]**) permettent d'autoriser le **GC** à supprimer l'objet référencé si le besoin s'en fait sentir. Ainsi, si la charge mémoire est importante, il est garantit que toutes les *références douces* seront libérées **AVANT** qu'une **OutOfMemoryException** ne soit lancée. Cela permet de conserver en mémoire de gros objet afin d'éviter d'avoir à les recharger plus tard, tout en permettant au **GC** de les supprimer en cas de besoin...
Les *références douces* peuvent être libérer par le **GC** si aucune *référence forte* n'existe pour l'objet en question. Toutefois, les implémentations des JVM sont encouragées à ne pas supprimer les *références douces* récemment créées ou récemment utilisées...

Exemple d'utilisation : Gérer un cache d'objet

Bien entendu, la gestion de cache d'objet est le cas typique d'utilisation de **SoftReference**. En effet, imaginez une application qui utiliserait une (ou plusieurs) image(s) de taille conséquente à divers moments de son exécution. Si on conserve constamment une *référence forte* sur cette image, elle peut se retrouver avec des problèmes de mémoire alors que l'image n'est pas utilisée.
Avec une *référence douce*, l'image serait supprimé par le GC en cas de besoin, et il ne nous restera plus qu'à la recharger le cas échéant. Mais si le besoin de mémoire ne se fait pas sentir, l'image restera chargé et cela évitera ainsi de la recharger...
Par exemple, on pourrait utiliser le code suivant :

```
SoftReference<Image> softImage =
  new SoftReference<Image>(null);
    public Image getImage() throws IOException {
    Image image = this.softImage.get();
    // Si la référence douce n'est pas valide
    if (image==null) {
      // On charge/recharge l'image
      image = ImageIO.read(filename);
      // Et on recrée une référence soft :
      this.softImage = new SoftReference<Image>(image);
    }
    return image;
  }
```

On peut bien sûr imaginer des systèmes de cache plus complexe en y combinant une **Map** ou une **List** qui se modifierait en conséquence (en utilisant une **ReferenceQueue** pour être tenu informé de l'invalidité des références)...
Selon un des commentaires de l'article anglais, la libération des *références douces* varie selon qu'on utilise la JVM **client** ou **server** :

- La JVM **client** tentera d'utiliser le moins de mémoire possible en supprimant les *références douces* avant d'augmenter la taille du **heap** (mémoire réservée).
- A l'inverse la JVM **server** tentera d'optimiser les performances en augmentant la taille du **heap** avant de supprimer les *références douces*.

Références faibles (*Weak references*)

Les références faibles (représentées par la classe **WeakReference[Lien30]**) permettent de stocker une référence, mais ne permettent pas d'assurer a elle seule que l'objet stocké sera conservé en mémoire. Les références faibles permettent ainsi de faire dépendre la durée de vie d'un objet par rapport à la validité d'un autre objet.
Les *références faibles* peuvent être libérer par le GC si aucune *référence forte* ni aucune *références douces* n'existe pour l'objet en question.

Exemple d'utilisation : Associer des valeurs à un objet

Lorsque l'on souhaite associer une valeur à un objet, la meilleur solution est bien sûr de créer un attribut spécifique. Toutefois, dans certain cas cela n'est pas possible, soit parce que l'on ne peut pas étendre l'objet, soit parce qu'on reçoit une implémentation spécifique.
Prenons le cas d'une **Map** qui permettrait d'associer un identifiant à un **Process** lancé via la classe **Runtime**, pendant toute la durée de vie de ce dernier. Si on utilise une simple **HashMap**, il nous faudrait enlever explicitement l'objet de la **Map** lorsqu'on souhaite qu'il soit libérer, sinon notre **HashMap** conserve une référence forte sur cet objet qui empêche toute libération par le GC.
L'API propose pour cela la classe WeakHashMap[Lien31] qui permet de faire cela simplement. Elle utilise en effet une **WeakReference** sur les clefs (et seulement les clefs) qui lui sont passées. Ainsi on peut avoir le code suivant :

```
  Map<Process,String> map = new
WeakHashMap<Process,String>();

  Process process = Runtime.getRuntime().exec(...);
  map.put(process, "process_1");

  System.out.println( map.size() ); // affiche 1
  //...
  process = null;
  System.out.println( map.size() ); // peut afficher 0
```

Lorsque la référence forte *process* n'existe plus, l'objet peut être libéré par le GC, et la **WeakHashMap** s'adapte en supprimant le couple clef/valeur qui n'est plus d'aucune utilité (on n'a pas besoin de conserver l'identifiant d'un objet que l'on ne va plus utiliser). Bien entendu, cela dépend du passage du GC, et donc la taille ne varie pas instantanément...

Références fantômes (*Phantom references*)

Pour finir, les *références fantômes* (représentées par la classe **PhantomReference[Lien32]**) sont un peu particulière dans le sens où leur méthode **get()** renverra toujours **null**. En effet l'objectif des *références fantômes* est d'être avertis de la suppression complète d'un objet par le **GC**. En effet, contrairement aux **WeakReference** et **SoftReference** qui sont "invalidées" dès que commence le processus de libération, les **PhantomReferences** ne sont ajoutées à la **ReferenceQueue** associée que lorsque l'objet référencé est complètement supprimé de la mémoire (cela permet d'éviter des "résurrections obscures" d'objet dans la méthode **finalize()**).
Les *références fantômes* peuvent être libérer par le GC si aucune *référence forte*, aucune *références douces* ni aucune *référence faible* n'existe pour l'objet en question.

Exemple d'utilisation : Libérer la mémoire

En effet, un grand nombre de personnes pensent qu'il est impossible de savoir si un objet a été libéré ou pas de la mémoire. Or c'est exactement ce que permet les *références fantômes*.
Lorsqu'on travaille avec de gros objets, cela permet par exemple de laisser le temps au GC de libérer les objets inutiles avant d'en charger d'autres.
La référence est ajouté en queue lorsque l'objet est complètement effacé de la mémoire...

```
  // On crée un objet
  Object o = new Object();
  // On crée une queue et une référence fantôme pour cet objet
  ReferenceQueue<Object> queue = new ReferenceQueue<Object>();
  PhantomReference<Object> ref = new PhantomReference<Object>(o, queue);

  // On libère l'objet :
  o = null;
  // On force le GC :
  System.gc();
  // Et on attend que le GC libère vraiment l'objet
  try {
    queue.remove();
  } catch (InterruptedException e) {
    e.printStackTrace();
  }
```

En résumé

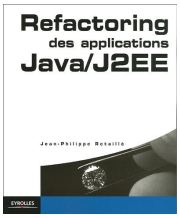
- Les *références douces* (**SoftReference**) permettent de gérer des caches en autorisant la suppression si nécessaire.
- Les *références faibles* (**WeakReference**) permettent de référencer des objets sans pour autant empêcher leurs suppressions.
- Les *références fantômes* (**PhantomReference**) permettent de s'assurer de la destruction d'un objet.

Avec tout cela, vous avez toutes les cartes en main pour gérer efficacement la mémoire de vos applications Java, mais attention toutefois à ne pas utiliser ces références de manière abusive... après tout le Garbage Collector fait assez bien son travail...

Retrouvez le billet d'adiguba en ligne : Lien33

Les Livres Java

Critiques de Livres Java



Refactoring des applications Java/J2EE

376 pages, Septembre 2005 Editions Eyrolles, ISBN: 2-212-11577-6

Public visé : Confirmé

Commandez sur Amazon.fr

Résumé de l'éditeur

Améliorer la qualité et l'évolutivité des applications Java/J2EE. Le refactoring consiste à refondre le code source d'une application existante ou en cours de développement pour en améliorer la qualité, avec pour objectif une réduction des coûts de maintenance et une meilleure évolutivité. L'ouvrage passe en revue les différentes techniques de refactoring utilisées en environnement Java/J2EE : extraction de méthodes, généralisation de type, introduction de design patterns, programmation orientée aspect, optimisation de l'accès aux données, etc. Un livre pratique illustré d'une étude de cas détaillée. L'ouvrage décrit dans le détail le processus de refactoring d'une application Java/J2EE : mise en place de l'infrastructure et des outils, analyse de la conception et du code de l'application, mise en oeuvre des techniques de refonte, tests de non régression. Cette démarche est illustrée par une étude de cas complète : refactoring d'une application J2EE Open Source à l'aide d'outils tels que Eclipse, CUS, JUnit et PMD.

Critique du livre par la rédaction (Stessy)
Quelle personne ne s'est jamais retrouvée en face d'un code qu'elle avait ou n'avait pas produit et qui parfois se révèle être une cochonnerie en terme de maintenance et d'évolutivité.
Eh bien moi je me suis retrouvé dans le cas ou ce code ne m'appartenait pas et, qui plus est, était mal agencé pour me permettre de le faire évoluer sereinement sans devoir tout réécrire de A à Z.
Je me suis donc retroussé les manches et ai commencé à proprement parler le refactoring de l'application.
Je dois bien avouer que je ne m'en serait jamais sorti si je n'avais pas suivi les conseils précieux distillés dans cet ouvrage.
L'auteur commence d'abord par expliquer le pourquoi du refactoring.
Ensuite viennent les pré-processus indispensables pour effectuer un refactoring robuste tels que le versioning des différentes ressources, la mise en oeuvre des tests unitaires,...
Le troisième chapitre est pour moi le plus théorique. Il aborde les méthodes qui permettent d'analyser le code tant de manière quantitative que qualitative
Je dois bien avouer que j'ai du relire ce chapitre une seconde fois pour bien comprendre à quels moments, et quels endroits du code, le refactoring devenait une nécéssité voire même une obligation.
Chapitre 4. Là, on rentre réellement dans la mise en oeuvre du refactoring. Chaque technique de refactoring abordée dans ce chapitre est basée sur un plan bien précis. Ce plan est décomposé en 4 parties :

- les objectifs de cette technique ainsi que les risques pouvant survenir
- les moyens de détection
- le refactoring proprement dit
- un exemple concret de mise en oeuvre

Le chapitre suivant traite de l'utilisation de test unitaires en combinaison avec le refactoring. Il explique également comment créer des simulacres d'objets qui permettront, comme le terme l'indique, de simuler des objets.
C'en est terminé avec la première partie de l'ouvrage.

La seconde partie traite des techniques avancées pour effectuer le refactoring d'une application telles que les design patterns, SOA.

Ce qui m'a le plus surpris dans cette partie, c'est le refactoring de la base de données.
Comme quoi on en apprend tous les jours.

La dernière partie reprend une étude de cas complète. Cette partie de l'ouvrage met en pratique la totalité des techniques abordées lors des chapitres précédents, en passant :

- par l'analyse quantitative et qualitative,
- par le refactoring de l'application,
- par la mise en oeuvre des tests unitaires

En conclusion, cet ouvrage est un petite merveille. A recommander chaudement.

Retrouvrez les critiques du livre en ligne: [Lien34]

Spring par la pratique

518 pages, Avril 2006 Editions Eyrolles, ISBN: 2-212-11710-8

Public visé : Tous niveaux

Commandez sur Amazon.fr

Résumé de l'éditeur

Simplifier le développement des applications Java/J2EE.
Cet ouvrage montre comment développer des applications Java/J2EE professionnelles et performantes grâce à Spring, associé à d'autres frameworks populaires telles que Struts, Hibernate ou Axis. Spring s'appuie sur des concepts modernes, tels que la notion de conteneur léger, l'inversion de contrôle ou la programmation orientée aspect, afin d'améliorer l'architecture des applications Java/J2EE en les rendant plus souples, plus rapides à développer et plus facilement testables.
Un livre pratique illustré d'une étude de cas détaillée.
L'ouvrage présente les concepts sur lesquels reposent Spring avant de détailler les différentes facettes du développement d'applications Web avec Spring : couche présentation (Struts, Spring MVC, Spring Web Flow, portlets, applications Ajax), persistance des données et gestion des transactions, intégration avec d'autres applications et sécurité applicative.
L'accent est mis tout particulièrement sur les bonnes pratiques de conception et de développement, qui sont illustrées à travers une étude de cas détaillée, le projet Open Source Tudu Lists.

Critique du livre par la rédaction (Christophe Jollivet)

Vous avez sans doute entendu parler de Spring, mais vous vous demandez ce que c'est, ce que cela peut faire pour vous? Alors ce livre est pour vous.

Ce livre présente un tour d'horizon du framework Spring. Mais il fait plus que de vous donner une idée, il vous montre comment l'utiliser. Si ce livre s'appelle "Spring par la pratique" c'est qu'il s'appuie entièrement sur une application écrite à l'occasion : TUDU List. Vous pouvez voir cette application sur le site : http://app.ess.ch/tudu/welcome.action . Différentes versions de cette application ont aussi été créées correspondant aux différentes technologies présentées dans le livre.

La première partie du livre commence par présenter les problématiques de la programmation Java et Java EE (séparation de préoccupation, test, indépendance de la plateforme et du serveur...). Ensuite il présente brièvement les réponses qu'apporte Spring avec la notion de conteneur léger et de POA. Enfin il présente brièvement l'application TUDU List qui sert de fil rouge au livre.

On trouve ensuite 4 chapitres très denses en informations. Le premier revient en détail sur les conteneurs léger et les notions d'inversion de control et d'injection de dépendance. Le second présente Spring en tant que conteneur léger et son utilisation. Le troisième présente la POA et enfin le quatrième l'intégration de la POA dans Spring.

Ensuite le livre aborde les différentes technologies et frameworks utilisés en Java EE et reprend la même structure tout du long : Présentation sommaire de la technologie Intégration dans Spring Intégration dans TUDU List Par exemple dans le cas de Struts, les auteurs commencent par présenter le framework en définissant MVC2, les actions et les FormBeans, ensuite ils présentent les trois possibilités d'intégration de Struts dans Spring, enfin ils présentent la solution utilisée dans TUDU List avec des morceaux de code et de fichiers de configuration.

Le livre aborde donc successivement tous les aspects : framework de présentation (Struts, Spring MVC, Spring WebFlow, Ajax et Portlets), gestion des données (persistance, transactions), l'intégration (JMS, JCA, XML, Acegy Security) et enfin les outils connexes (supervision avec JMX et test).

Même si certains points mériteraient d'être plus détaillés, ce livre est indispensable pour qui veut un aperçu complet de ce que peut lui apporter Spring et de ce que va lui coûter sa mise en oeuvre. On peut déplorer que parfois les extraits de code soient trop courts pour permettre de bien comprendre où ils se situent et nécessitent de se reporter aux codes sources de l'application. Cela rend obligatoire le fait d'avoir un ordinateur sous la main, ou de laisser des commentaires si l'on est coincé dans un train.

Retrouvrez les critiques du livre en ligne: [Lien35]

Liens

Lien1 : http://www.jcs.be/
Lien2 : http://javapolis.com
Lien3 : http://www.jetbrains.com/idea/
Lien4 : http://springframework.com/
Lien5 : http://www.hibernate.org/
Lien6 : http://www.caucho.com/
Lien7 : http://edocs.bea.com/wls/docs92/index.html
Lien8 : http://www.jboss.com/
Lien9 : http://maps.google.com/
Lien10 : http://java.developpez.com/annonces/open-sourcing-java/
Lien11 : http://java.developpez.com/annonces/open-sourcing-java/#LIII
Lien12 : http://java.developpez.com/interview/open-source/alexismp/
Lien13 : http://java.developpez.com/interview/open-source/gfx/
Lien14 : http://java.developpez.com/interview/open-source/vbrabant/
Lien15 : http://adiguba.developpez.com/tutoriels/java/6/
Lien16 : http://brabant.developpez.com/outils/netbeans/5.5
Lien17 : http://baptiste-wicht.developpez.com/outils/jetbrains/intellij/idea/6/
Lien18 : http://dejardin.developpez.com/outils/jetbrain/intellij/idea/6/
Lien19 : http://gfx.developpez.com/tutoriel/java/gc/
Lien20 : ftp://ftp-developpez.com/ericreboisson/tutoriel/java/xml/xstream/fichiers/articleXStream.zip
Lien21 : http://www.eclipse.org/
Lien22 : http://dist.codehaus.org/xstream/jars/xstream-1.2.jar
Lien23 : http://ericreboisson.developpez.com/tutoriel/java/xml/xstream/
Lien24 : http://blog.developpez.com/index.php?blog=50&title=les_references_faibles_sous_net&more=1&c=1&tb=1&pb=1
Lien25 : http://weblogs.java.net/blog/enicholas/archive/2006/05/understanding_w.html
Lien26 : http://java.sun.com/j2se/1.5.0/docs/api/java/lang/ref/package-summary.html
Lien27 : http://java.sun.com/j2se/1.5.0/docs/api/java/lang/ref/Reference.html
Lien28 : http://java.sun.com/j2se/1.5.0/docs/api/java/lang/ref/ReferenceQueue.html
Lien29 : http://java.sun.com/j2se/1.5.0/docs/api/java/lang/ref/SoftReference.html
Lien30 : http://java.sun.com/j2se/1.5.0/docs/api/java/lang/ref/WeakReference.html
Lien31 : http://java.sun.com/j2se/1.5.0/docs/api/java/util/WeakHashMap.html
Lien32 : http://java.sun.com/j2se/1.5.0/docs/api/java/lang/ref/PhantomReference.html
Lien33 : http://blog.developpez.com/index.php?blog=51&title=comprendre_les_references_en_java&more=1&c=1&tb=1&pb=1
Lien34 : http://java.developpez.com/livres/?page=Francais#L2212115776
Lien35 : http://java.developpez.com/livres/?page=Francais#L2212117108

Rejoignez-nous!

Java.developpez.com, c'est

- **Une équipe de bénévoles à votre disposition**
  - **11 responsables** de rubriques et FAQs
  - **24 rédacteurs et modérateurs** réguliers
  - **45 contributeurs** occasionnels
- **136 codes source** mis à votre disposition par 29 auteurs
- **938 Q/R** réparties dans 11 FAQs
- **40 critiques** de livres
- des vidéos des événements Java
- des interviews
- des compte rendu
- **14 blogs** tenus par des membres de l'équipe, tous disponible depuis le récapitulatif Java.
- **+ de 200 cours et tutoriels**
- **41 sous-forums** découpés en 5 grandes familles
- des sessions de chats
- des débats
- des sondages
- ...

Java.developpez.com, c'est aussi

- Un Virtual Java User Group
- Un partenaire pour vos évènements Java

Java.developpez.com, c'est encore

Une équipe intégrée dans developpez.com, où vous trouverez également des débats, sondages, vidéos, critiques de livres, un magazine, une lettre d'information, des forums d'entre-aide, l'actualité, couvrant une multitude d'autres sujets comme la conception, la gestion de projets, xml, uml, html, css, ajax, webmarketing, sql, cvs, subversion, ...

C'est également d'autres langages, comme PHP, Delphi, Visual Basic, Python, DOTNET, C, C++, C#, Pascal, Assembleur, Flash, ...
C'est aussi des plateformes comme Windows, Linux, Unix
Mais également des bases de données, comme MySQL, Oracle, 4D, SQL Server, DB2, Interbase, PostgreSQL, Sybase, ...

Java.developpez.com, c'est SURTOUT

VOUS qui venez consulter nos FAQS, nos critiques de livres, nos blogs, nos cours, tutoriels, ...
VOUS qui postez sur nos forums, qui proposer des articles, des sources, des critiques, ...
Sans vous, java.developpez.com n'existerait pas

Rejoignez-nous

- Que vous soyez étudiant, développeur, analyste, architecte, gestionnaire de projets, vous trouverez certainement des réponses à vos question, de l'aide dans les forums,
- Vous écrivez des cours et tutoriels ? Nous les hébergeons gratuitement sur developpez.com !
- Vous désirez un blog pour écrire des billets en rapport avec le développement ou la programmation ? Nous l'hébergeons gratuitement sur developpez.com !
- Vous êtes enseignants ? Nous vous invitons à mettre vos cours à la disposition de tous
- Vous connaissez l'anglais ? Aidez nous à traduire des articles ou livres en français