Pourquoi utiliser Java ?

par

Date de publication :

Dernière mise à jour :

Java souffre toujours de sa mauvaise réputation, héritée de ses premiers jours. Nous allons voir que Java est aujourd'hui plus que jamais une solution toute à fait viable et de qualité pour le développement, même à hautes performances comme les jeux.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

1 - Qu'est-ce que Java ?

Java est un langage de programmation orienté objet et un environnement d'éxecution, développé par Sun Microsystems. Il fut présenté officiellement en 1995. Le Java était à la base un langage pour Internet, pour pouvoir rendre plus dynamiques les pages (tout comme le JavaScript aujourd'hui). Mais le Java a beaucoup évolué et est devenu un langage de programmation très puissant permettant de presque tout faire, je dis bien presque car nous verrons pourquoi il ne permet pas de tout faire. Java est aujourd'hui officiellement supporté par Sun, mais certaines entreprises comme IBM font beaucoup pour Java.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

2 - Java et JavaScript

Une erreur que l'on voit souvent sur les forums est la confusion entre Java et JavaScript en croyant que c'est le même et unique langage. C'est totalement faux ! Java est langage de programmation très évolué et puissant, qui, pour faire simple pour l'instant, doit être compilé, alors que le JavaScript est un langage de script pour les pages internet, qui est interprété par le navigateur. Il ne permet donc pas de faire autant de choses que le Java. Malgré son nom JavaScript, a été initiallement créé par Netscape. Mais sachez que le JavaScript, s'appele dans sa forme standardisée "ECMAScript" (l'ECMA est un organisme de standardisation) et dans sa forme Microsoftienne JScript.

Cependant Java 6.0, alias Java Mustang, sera accompagné d'un langage de script, ***Rhino Javascript****, qui vous permettra d'utiliser du Javascript dans vos codes Java.*

Plus d'informations à ces adresses : Javascript n'a rien à voir avec Java[...] par vbrabant et Rhino JavaScript sur le site de Mozilla.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

3 - Comment fonctionne Java ?

Java a un fonctionnement particulier, mais avant, voyons comment fonctionnent d'autres langages. Prenons par exemple le C, langage très connu et utilisé, le noyau Linux est d'ailleurs codé en C. On programme donc en C en écrivant le code, appelé code source, dans son éditeur. Mais la machine ne peut pas l'interpréter tel quel... Il faut donc le transformer en code machine compréhensible par votre PC. On utilise donc un compilateur qui va faire de ce code source composé de lettres, accolades, parenthèses, chiffres... une suite de 0 et de 1, c'est à dire un code binaire, compréhensible directement par votre processeur. C'est donc aussi pour cela qu'un programme compilé sur Windows avec un processeur Intel Pentium ne fonctionnera pas sur un Mac OS avec un processeur G4 par exemple.

Donc ce code C est en gros, le même sur n'importe quel OS, pour un code simple sans parties spécifiques, mais ce qui sera produit comme code binaire (suite de 0 et 1) ne sera pas le même. La portabilité (compatibilité entre les OS et architectures matérielles) est donc très faible : code source pas toujours réutilisable tel quel et obligation de recompiler le code sur chaque architecture sur laquelle le programme devra tourner. Prenons un autre langage, plus ancien et existant sous de nombreuses formes : l'assembleur. C'est le langage le plus proche de la machine. Il doit aussi être compilé, mais comme le langage est déjà très proche de celui de la machine, le programme tournera plus rapidement. Mais avec ce langage, la portabilité est nulle. Prenons maintenant un langage comme l'HTML. Cette fois-c c'est complètement différent. Le code ne sera pas compilé en langage binaire. Il sera directement interpreté par un logiciel, généralement un naviguateur (Firefox, Opera, Internet Explorer...). La compatibilité est parfaite même si on parle des fois de sites incompatibles, ce n'est pas la même "compatibilité". Votre code fonctionnera quelque soit l'architecture ou l'OS.

Maintenant voyons comment fonctionne le Java. Vous tapez donc votre code source dans votre éditeur préféré. Donc quand vous avez écrit votre programme, vous avez un ou plusieurs fichiers de codes sources. Mais la machine, tout comme pour le C et l'assembleur, ne peut pas l'exécuter tel quel. Il faut donc le compiler, mais c'est ici que Java diffère des autres langages. Je vais essaier de faire simple : il y a d'abord une première compilation qui va transformer votre code source en Java Byte Code. C'est un code, très proche du langage machine mais pas à 100%, donc il n'est toujours pas exécutable. Java est composé d'une JVM (Machine Virtuelle Java). Cette JVM est différente selon chaque système d'exploitation. Il y en a pour presque tous les systèmes d'exploitation, pour les téléphones portables, pour les PDA, Pocket PC...Parfois il y en a plus d'une par système d'exploitation, comme pour Linux ou Windows, mais nous verrons ça dans l'article suivant. Retenons que la JVM officielle, la plus utilisée, la plus à jour par rapport aux spécifications du langage Java est forcement celle de Sun, les propriétaires de Java. Donc revenons à notre Java Byte Code. Ce dernier va donc être "interprété" par la JVM en temps réel. Donc le code n'est pas directement executé par la machine, mais par une machine virtuelle. Nous verrons un peu plus loin qu'il y a encore une autre étape présente dans les nouvelles versions de Java et que cette étape change tout.

Write once, run everywhere

Ecrivez une fois, exécutez partout

C'est le slogan de Java. Maintenant souvenez vous de notre histoire avec la portabilité, avec par exemple le C qui devait souvent être réecrit plusieurs fois pour être adapté à chaque OS et compilé sur chaque plateformes... Avec Java, dans la majorité des cas, vous n'écrivez qu'une seule fois votre code, et vous ne le compilez (en Java Byte Code) qu'une seule et unique fois, sur votre machine par exemple. Imaginons que vous ayez compilé votre programme sur Windows avec un processeur Intel. Il sera compatible avec Linux et avec Mac OS X tournant sur un G4, ou même sur votre téléphone portable.e

Mais ce n'est pas toujours le cas. Trois choses peuvent faire perdre cette compatibilité :

- Le programmeur lui même : en codant avec les pieds...
- L'utilisation d'une API (une API est un ensemble de fonctions de programmations, ce sont des librairies de

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

code) spécifique : certaines APIs non-officielles peuvent ne pas être compatibles avec tous les OS, pour diverses raisons. L'API non compatible bridera donc votre portabilité. Mais une API compatible avec un seul OS est très rare. Votre programme sera au moins supporté sous Windows, Linux et à moindre mesure, Mac OS X. Mais comme Java, pour garder une compatibilité parfaite, se limite généralement au plus petit dénominateur commun, il faut parfois avoir recours à des API tierces pour avoir accès à des fonctions plus spécifiques. Mais ceci limite bien entendu la compatibilité entre les systèmes.
- La JVM : certaines JVM non-officielles ne supportent pas toutes les fonctionnalités de Java. D'autres JVM faites spécialement pour l'embarquée (téléphones portables, PDA, Palm, Pocket PC...) ne peuvent pas implémenter toutes les fonctionnalités à cause du matérielle, par exemple la 3D ne sera pas implémentée sur un téléphone portable (sauf récemment, avec de nouvelles JSR en place, pour la 3D accélerée sous les portables nouvelle génération).È

Mais à part ça, nous aurons une compatibilité parfaite. Maintenant prenons le cas de la 3D. Cette fois-ci, cela dépend surtout de l'implémentation d'OpenGL... Et les APIs pour accéder à OpenGL en Java ne sont pas toujours compatibles avec tous les OS. Les problèmes de compatibilités dans ce domaine seront similaires dans tous les langages, mais en principe, nos codes seront compatibles avec les plateformes Windows, Linux et Mac OS X, c'est à dire l'écrasante majorité des utilisateurs.

De plus des JSR sont en préparation pour l'utilisation d'OpenGL sur les portables de nouvelle génération. Encore un point pour Java.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

4 - Java est-il si lent que l'on dit ?

C'est une rumeur qui n'a plus lieu d'être. Les programmes utilisant les premières versions de la JVM 000taient entre 20 et 40 fois plus lents que leur équivalents en C par exemple. Tout simplement car le Java Byte Code était interpreté par la machine virtuelle. Or par définition, une code interpreté est plus lent qu'un code natif. Mais c'était en 1993...A partir de 1995, Sun a incorporé la technologie HotSpot, un compilateur JIT (Just-In-Time) dans ses JVM. Il s'agit en fait d'une autre étape de la compilation, mais dont vous n'avez pas à vous occuper.

Vous connaissez certainement la loi des 20/80 de l'économiste italien Vilfredo Pareto qui dit que 80% des richesses sont détenus par 20% de la population. Eh bien ce principe est applicable en programmation, que l'on pourrait formuler de la manière suivante :

Seul 20% du code nécessite 80% des ressources de la machine.-

La JVM repère les morceaux de code très utilisés et les compile en code natif, donc qui ne passe plus par une interpretation et qui est donc forcement plus rapide. Cette technologie devient de plus en plus performante à chaque nouvelle version. De plus vous pouvez lancement une compilation complète de l'application en code natif, donc il n'y aura plus d'interpretation de code. Cependant le temps de lancement sera bien plus long.

De nos jours, sur une machine correcte, une programme Java est en moyenne entre 1.1 et 1.3 plus lent que son équivalent en C++ mais ce ne sont que des chiffres. Dans certains cas, et de plus en plus souvent, le Java éxecutés dans une JVM récente a les mêmes performances que les autres langages compilés en natifs. Certains benchmarks (tests de performances) montrent même que sur les 3/4 des tests, la JVM 5.0 serveur de Sun est beaucoup plus rapide qu'un code en C++.

Mais pourquoi cette différence de rapidité entre Java et un langage compilé ? Cela s'explique par le fait que HotSpot effectue une compilation dite de Runtime, c'est à dire qu'elle est effectuée au lancement de l'application. Quand vous compilez un code en C par exemple, vous pouvez spécifier que le compilateur doit faire des optimisations pour une certaines architectures, mais cela reste encore assez générique, car vous ne connaissez pas la machine sur laquelle va tourner l'application. Or avec une compilation au lancement de l'application, on connait tous les détails techniques de la machine, plus la configuration du système. Cela permet donc à HotSpot d'effectuer des optimisations beaucoup plus poussées, qu'aucun compilateur natif ne peut faire. Voilà pourquoi il arrive d'avoir des performances meilleurs en Java que dans d'autres langages comme le C++.

Mais retenez bien qu'avec cette technologie, une application Java devient vraiment rapide qu'après un temps relatif d'utilisation (sauf si vous compilez tout au début) et que généralement la seconde exécution d'un programme est plus rapide que la première.

Malheuresement ce temps de lancement est toujours plus long que pour les autres langages car il faut lancer la JVM en même temps, et qui va effectuer de nombreuses optimisations. Même si la dernière versions à augmenter de 30% la rapidité de lancement et que Java 6.0, encore en développement, a réecrit certaine partie comme le Java Verifier, pour un temps de lancement encore plus court, un splash screen (écran de démarrage) est toujours recommandé pour vos applications de taille moyenne à grande.

Java consomme donc plus de RAM que les autres langages, même pour un petit programme. C'est pour ça que généralement plus de 256Mo de RAM sont recommandés mais on peut bien sûr faire avec sans trop de problèmes avec beaucoup moins de RAM. De toute façon de nos jours, 256 Mo est souvent trop peu et peut brider le reste du PC si il est puissant. De plus, il existe deux versions officielles de la JVM de Sun, mais c'est transparent pour l'utilisateur, et elles seront sûrement dans le future, réunies en une seule. Il y a une JVM "client", plus rapide à

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.
http://info-rital.developpez.com/

lancer, qui consomme moins de ressource, mais moins performante, c'est celle là qui est lancée par défaut et une JVM "server", plus longue à démarrer, qui consomme plus de RAM, mais qui est beaucoup plus rapide. Cela s'explique par le fait que la JVM "server" effectue bien plus d'optimisations.

Mais de toute façon, la rapidité du programme dépendra principalement du développeur. On peut très bien faire un programme très rapide en Java, et en réaliser un beaucoup plus lent en C. La principale optimisation vient du développeur et des algorithmes choisis.

De plus les accusations sur les performances de Java viennent généralement de personnes développant principalement dans d'autre langages. On ne développe pas en Java comme on le ferait en C. Certaines techniques et optimisations peuvent être très efficaces dans un code en C mais peuvent faire chuter les performances d'un programme Java.

En Java il est souvent conseillé d'éviter de faire certaines optimisations de bas niveau dans le code. Il vaut mieux laisser faire HotSpot et écrire un code clair et maintenable car certaines optimisations peuvent se réveler efficaces avec une telle version de Java mais peuvent devenir inutiles voir dangereuses pour une version suivante de Java.

Je vous renvoie sur le très bon article de adiGuba pour un benchmark détaillé entre la JVM de Sun et du code natif : La machine virtuelle Java est-elle vraiment lente ?.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

http://info-rital.developpez.com/

5 - Le haut niveau d'abstraction

Java et sa JVM permettent au programmeur d'avoir un très haut niveau d'abstraction par rapport à la machine. Il n'aura donc pas besoin de s'occuper de la compatibilité matérielle. Mais la JVM a aussi d'autres avantages : pour ceux qui connaissent le C, sachez qu'il n'y a plus ces infâmes (pas pour tous) pointeurs, qui peuvent être fatal si ils sont mal utilisés et surtout, la gestion de la mémorie est automatique. Le Garbage Collector se lance à un intervalle régulier, pour libérer la mémoire occupée par les objets qui ne sont plus du tout utilisés dans le programme. Ce Garbage Collector est bien entendu paramètrable. Attention, l'utilisation d'un Garbage Collector ne dispense pas d'écrire un code propre. D'ailleurs il existe certaines techniques avancées de programmation en Java pour une utilisation parfaite du Garbage Collector. Nous en verrons certainement dans d'autres articles. Un langage de haut niveau (d'abstraction) permet de développer plus rapidement, d'être bien plus productif sans avoir à se soucier des problèmes que j'ai cités avant.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

6 - La richesse de l'API Java

Contrairement à la plupart des autres langages (sauf la plateforme .Net), Java met à la disposition du développeur une API très riche lui permettant de faire de très nombreuses choses. Contrairement à des langages comme le C, où il fallait coder par nous même des fonctionnalités basiques comme le chargement d'images, ou bien d'avoir recours pour quasiment n'importe quoi à une librairie de code tierce, Java vous propose à peu près tout ce dont vous avez besoin directement dans le JDK. Ceci est un énorme avantage, qui augmente encore une fois grandement votre productivité de développement.

De plus il existe énormement d'API tierces de très bonnes qualités, pour des fonctionnalités qui viendraient à manquer au JDK.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

7 - Un langage orienté objet

Java est complètement orienté objet. Java vous permet et vous pousse même à développer vos applications d'une façon orientée objet et vous permet d'avoir une application bien structurée, modulable, maintenable beaucoup plus facilement et efficace. Cela augmente une fois de plus votre productivité.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

8 - Java, un langage ou une plateforme ?

Mais Java ne se limite pas à la plateforme développée par Sun. Il existe de nombreuses implémentations du JDK, de JVM ou même de compilateurs qui permettent d'obtenir des exécutables natifs, mais vous trouverez bien plus d'informations dans un article consacré à toutes ces implémentations et JVM.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

9 - Une amélioration constante

Prenez un code C compilé il y a 8 ans et lancez le aujourd'hui. Soit il se lance et a les mêmes bugs et performances qu'il y a 8 ans, soit il ne se lance pas du tout. Faites pareil avec un code Java compilé en Java Byte Code. Il se lancera très certainement mais cette fois-ci vous avez de grandes chances que le programme possède beaucoup moins de bugs et soit beaucoup plus performants. Java conserve une compatibilité assez bonne avec les anciens codes, compilés avec une version antérieur.

Java est en constante amélioration. Chaque nouvelle version apporte son lot de nouveautés au niveau de l'API et apporte de nettes améliorations au niveau des performances et de la stabilité. C'est pourquoi il faut comparer ce qui est comparable. Quand vous voulez parlez de Java, ne vous référez pas à un Java 1.3 ou même au JRE 1.1 de Microsoft incorporé par défaut dans tous les naviguateurs IE, car il ne s'agit pas du tout de la même chose. Alors pensez toujours à utiliser une version récente du JRE comme Java 5.0 ou même Java 6.0 encore en développement. Et si vous voulez aller plus loin, lancez la JVM server plutôt que la JVM client.

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.

10 - Conclusion

Nous avons donc vu que Java possède de nombreux avantages non négligeables.

Cependant nous avons droit au revers de la médaille. Souvenez-vous, au début je vous disais que Java permettait de presque tout faire. Notez bien le "presque". A cause de ce haut niveau d'asbtraction, Java ne permet pas d'accéder directement au matériel. C'est pour ça par exemple qu'il n'y a pas de driver en Java pur, ni d'application proche de la machine en Java. Cependant, grâce à JNI, le problème est plus ou moins résolu, même si une partie doit être écrite dans une langage ayant directement accès à l'OS et à la machine (C, C++...).

Mais bon, mis à part l'accès direct au GPU, qui est possible grâce à OpenGL et ses wrappers Java, est-ce vraiment un problème ?

- Portabilité excellente
- Langage puissant
- Langage orienté objet
- Langage de haut niveau
- JDK très riche
- Nombreuses librairies tierces
- Très grande productivité
- Applications plus sûres et stables
- Nombreuses implémentations, JVM et compilateurs, libres ou non
- IDE de très bonne qualité et libres : Eclipse et Netbeans par exemple
- Supporté par de nombreuses entreprises telles que Sun ou encore IBM et des projets comme Apache

Copyright © - F. De Leo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de domages et intérêts.