

François Tonic

OPENJDK

java

Le monde Java en pleine mutation : version LTS, support officiel, OpenJDK, Java EE...

Depuis 18 mois, le paysage Java s'est considérablement modifié. Oracle a reversé Java EE et divers composants liés à la fondation Eclipse. La roadmap a été entièrement modifiée pour accélérer les livraisons de versions, tout en lissant les fonctionnalités importantes dans le temps. Et le modèle économique de Java s'est trouvé bouleversé dans l'offre officielle d'Oracle. Pourquoi faire simple quand on peut donner un bon mal de tête aux développeurs ?

Ces changements, parfois radicaux, opérés par Oracle, répondent à plusieurs critères :

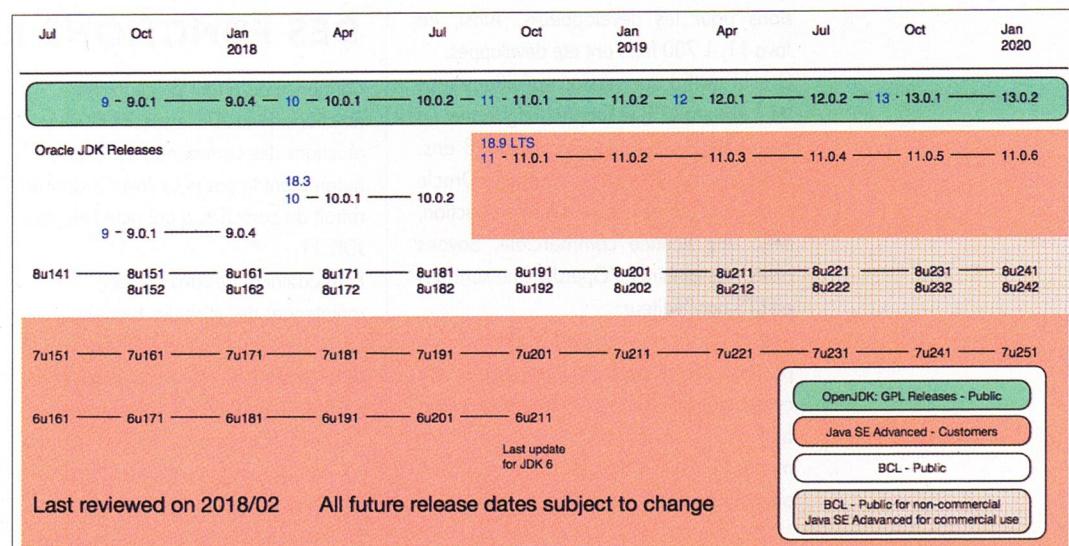
- Recentrer le développement sur quelques éléments clés ;
 - Raccourcir les délais de déploiement des versions, ne plus faire de big bang à chaque version majeure ;
 - Définir clairement les licences commerciales et open source ;
 - Avoir un modèle de tarification simplifiée.
- Bernard Traversat (VP Engineering, Oracle) a apporté des précisions sur la démarche d'Oracle : quel investissement mettre pour faire avancer la plateforme ? Faut-il dépenser des \$ et des efforts pour maintenir les anciennes versions au détriment des évolutions ? L'éditeur ne veut pas, ou ne peut pas, développer l'ensemble de Java. Le fait de confier plusieurs gros projets (Java EE, Glassfish, NetBeans) à des fondations open source répond à cette nouvelle stratégie, tout en recentrant le développement et les efforts sur le « Java Core », le cœur même de Java avec la JDK.

Trois coûts sont décrits par Bernard Traversat :

- comment la fonction va être compatible avec le reste de Java, sans provoquer de casse ,
- le coût de l'implémentation ,
- s'assurer que les choix ne seront pas bloquants dans le futur.

Roadmap : une nouvelle cadence de sortie et version LTS

Depuis le rachat de Sun par Oracle, Java continuait à évoluer, mais souvent avec des retards et des coupures fonctionnelles. L'éditeur avait annoncé des roadmaps am-



bitieuses jusqu'en 2022 dont la réalité faisait douter. Cette ambition s'est heurtée à la réalité : des retards dépassant 12 mois pour certaines versions.

2017, Oracle bouleverse nos habitudes en annonçant une version tous les 6 mois de Java SE, en ne parlant plus forcément de version majeure, mais de version fonctionnelle. L'avantage de cette évolution est de se focaliser sur des évolutions fonctionnelles sans attendre plusieurs années pour en bénéficier. Pour les éditeurs d'outils, l'idée est de pouvoir intégrer plus rapidement ces nouvelles versions en minimisant les changements. Enfin, cela évite l'effet « j'suis déçu, y'a rien de nouveau » ou « y'a trop de nouveautés ». Bref, on lisse les évolutions. Les versions de maintenance, patchs, etc. sortiront toujours. Ces versions « 6 mois » sortiront en mars et septembre. Il s'agit d'accélérer l'adoption par les développeurs des nouvelles versions et d'éviter

OU VA JAVA ?

En prenant du recul, les mouvements actuels prennent du sens. La cadence imposée de 6 mois est peut-être la force et la faiblesse de la nouvelle stratégie. Le rythme nous paraît trop élevé, même si éviter le big bang des versions majeures est indispensable, mais aussi les multiples retards de ces versions. Nous pourrons tirer un premier bilan dans les 12 à 18 mois.

C'est un moyen de redynamiser le langage face à une concurrence nouvelle et féroce : JavaScript, Go, Swift, Rust, Python, etc. Ils apparaissent comme des langages plus dynamiques et plus récents. Il faut avouer que Java n'est plus un langage qui donne envie, et son côté verbeux, lourd et lent (malgré tous les efforts), sont des critiques récurrentes. Il est urgent de dépasser Java.

Mais une question demeure : faut-il continuer à miser sur Java ? Ne va-t-il pas devenir le nouveau Cobol ?

d'attendre x mois, voire des années pour passer à une version plus récente. Bernard Traversat nous précise que cette nouvelle cadence doit permettre d'ajouter des fonctions en continu, sans attendre plusieurs années. Un des problèmes de l'ancienne stratégie des versions majeures tous les 2 à 3 ans est de retarder le déploiement de fonctions et d'améliorations prêtes ; il fallait attendre le développement de toutes les fonctions prévues pour sortir la release. Le rythme de 6 mois casse cette logique de big bang tout en réduisant les risques d'incompatibilités et doit permettre d'accélérer l'adoption des nouvelles versions pour les développeurs. Ainsi, en Java 11, 1 700 fixes ont été développés.

Oracle sortira une version spécifique tous les 3 ans : la LTS, le support long terme. La Java LTS d'Oracle sera supportée 8 ans. Cette version sera proposée par Oracle seul. La LTS sera destinée à la production, avec une licence commerciale. Soyons clairs : les binaires d'OpenJDK ne sont pas en LTS par l'éditeur.

Selon Oracle, les versions LTS doivent être plus stables et correspondent mieux aux cycles longs de certaines infrastructures et applications. Java ne fait que reprendre le modèle LTS utilisé depuis longtemps par les principales distributions Linux.

Oracle JDK vs OpenJDK

Deux JDK sont à distinguer : Oracle JDK, la version officielle de l'éditeur et OpenJDK, la version open source de Java. Cette dernière est mise en licence GPL. Oracle JDK est sous licence Oracle Binary Code Licence Agreement for Java SE.

Pour Oracle, il ne s'agit pas d'avoir une différence fonctionnelle, mais d'avoir une harmonisation entre les deux JDK : rajouter les fonctions développées dans la version Oracle dans OpenJDK ou encore supprimer les fonctions comme Java FX retirée dans l'Oracle JDK.

Oracle a précisé dès septembre 2017 que :

- OpenJDK est une version gratuite disponible pour les développeurs, les entreprises, sans support officiel d'Oracle.
- Oracle JDK : version LTS avec support niveau entreprise, pour la production. Cette JDK est une version commerciale.

Si vous faites des développements déployés en production ou des apps vendues utilisant

Oracle JDK vous devrez payer la licence pour l'utiliser. Cette JDK est utilisable en usage non commercial. Dans ce cas, vous n'avez pas besoin d'une licence. Les mises à jour (update) gratuites seront limitées dans le temps. Pour accéder aux updates Oracle, il faudra être client. C'est donc la fin des updates publiques. Java 8, sortie en 2014, aura des updates jusqu'à fin 2020. Les souscripteurs Java SE d'Oracle auront un support étendu de plusieurs années supplémentaires. L'éditeur encourage la migration de Java 8 vers une

version plus récente. Si vous n'avez pas besoin de support officiel d'Oracle ni des mises à jour de l'éditeur, installez OpenJDK. Si vous cherchez la JDK depuis un moteur de recherche, vérifiez bien que vous installez une version open source et non la version Oracle. Il existe plusieurs JDK gratuites basées sur OpenJDK. Tout un écosystème de builds s'est bâti autour d'OpenJDK. Ces versions doivent passer les tests TCK pour pouvoir être pleinement compatibles et surtout pour rassurer les entreprises et développeurs. Le passage du

DES FONCTIONS RETIRÉES

L'annonce du retrait de certaines fonctions de la JDK a suscité de vives réactions des communautés. C'est notamment le cas pour JavaFX dont le retrait du core JDK a été acté avec la JDK 11.

La modularité de Java permet maintenant de retirer les modules souhaités ou de les faire évoluer différemment. C'est le cas de JavaFX. « Il y avait un intérêt à faire évoluer JavaFX différemment du Core ». Ainsi, FX aura son propre rythme d'évolutions et la nouvelle version pourrait être incluse dans une future version de la JDK. Cependant, JavaFX subit la domination de HTML 5, CSS, JavaScript. Les technologies concurrentes (Silverlight, Flash) ont fini par être abandonnées. Le support est assuré pour plusieurs années. Mais le fait que la pile ne soit plus un standard va aussi inciter les développeurs à voir d'autres solutions.

Au printemps 2018, Oracle avait, en plus de JavaFX, annoncé plusieurs changements :

- Support de Web Start application de Java 8, mais non inclus dans Java 11. Oracle encourage le retrait de cette fonctionnalité ;
- Évaluation sur les évolutions et le support de AWT et Swing ;
- Dépréciation des applets : avec la suppression du support des applets dans les navigateurs, l'éditeur a déprécié ce module dès Java 9 avant son retrait dans Java 11.

L'avenir de Java ME

Le cycle de Java ME (et Java ME Embedded) est déconnecté de la JDK. La dernière version remonte à janvier dernier avec Java ME SDK 8.3.1 incluant le runtime Java ME Embedded 8.3. Cette version est une simple version de maintenance et limitée à Windows. Depuis, les informations sur l'avenir de cette plateforme sont rares.

Oui, Java ME est toujours dans le giron d'Oracle et des discussions se déroulent dans le JCP pour son avenir. Mais avec la modularité de Java, l'apparition d'un véritable core, Java ME a-t-il un intérêt ? Pour Bernard Traversat, Java est désormais proche de ME. Faut-il garder cette édition ou la remplacer par autre chose ?

Il y a tout de même urgence à clarifier la situation : abandonner purement et simplement, reverser à une fondation, ou rebooster la plateforme en y mettant les ressources nécessaires, notamment sur la partie IoT où la concurrence est féroce.

Continuer à soutenir les langages tiers dans la JVM

Bernard Traversat nous a réaffirmé l'importance des langages supportés par la JVM : Groovy, Scala, Python, Closure, etc. L'effort ne sera pas coupé et l'objectif est d'assurer le meilleur support à ces langages.

- Minimal syntax, maximum sugar

```
record Person(String forename, String surname) {};  
  
// creates fields  
// creates getters - forename(), surname()  
// creates equals/hashCode/toString  
// creates constructor/deconstructor
```

TCK permet d'assurer la compatibilité et d'obtenir la licence officielle. A noter que le TCK côté Java SE n'est pas open source comme c'est désormais le cas pour l'édition dédiée à Java EE.

Les versions Oracle et OpenJDK sont quasi-méthodes identiques à quelques détails. Certaines fonctions de Java 9 étaient orientées entreprises avec un modèle commercial. Ces éléments ne se retrouvaient pas dans OpenJDK et pouvaient provoquer des erreurs. Les deux JDK sont identiques à 99,9 %.

Cependant, la réalité d'aujourd'hui n'est pas forcément la réalité de demain.

Oracle Java SE : on passe à la caisse

La version d'Oracle est donc une version commerciale de Java SE. Il s'agit de proposer une souscription, et non plus une licence perpétuelle comme c'était le cas précédemment. Deux tarifications sont proposées :

- Partie Java Desktop : 2,5 \$ par utilisateur et par mois ;
- Partie serveur : 25 \$ par processeur physique et par mois.

Les tarifs sont fortement dégressifs si vous avez des centaines de processeurs ou des milliers d'utilisateurs. Sur la partie serveur, il s'agit bien de processeur, quel que soit le nombre de coeurs. Oracle nous a confirmé cet élément. La souscription permet d'accéder aux supports techniques 24/7, aux mises à jour, etc.

NetBeans

L'IDE NetBeans, un des outils historiques du monde Java, a connu une histoire mouvementée. L'outil naît en 1997. Il est racheté par Sun en 1999. L'année suivante, il est mis en open source. En 2016, Oracle cherche à se délester de l'IDE. En automne

de la même année, le projet rejoint la fondation Apache.

Il a fallu 2 ans d'efforts pour sortir une première version majeure de l'IDE, la v9, depuis la sortie de la 8.2 (octobre 2016). Ce long délai est dû à la migration du code, aux changements d'équipes, les process qui ne sont pas les mêmes, etc. Et le chantier n'est pas encore totalement terminé. NetBeans représente tout de même 7,4 millions de lignes de code. C'est le plus gros projet actuellement géré par la fondation.

La compétition est sévère entre les IDE Java. Aujourd'hui, IntelliJ de JetBrains est souvent en tête des préférences des développeurs. Eclipse est lui aussi très présent, notamment dans les solutions de développement utilisant Eclipse comme fondation technique. NetBeans est derrière ce duo, même si l'outil conserve une solide base de fidèles. Mais il faut avouer que les formations, les écoles ou encore les éditeurs supportent plus facilement Eclipse que NetBeans. Eclipse a pour lui un écosystème très riche, avec des centaines de plug-ins et une couverture fonctionnelle particulièrement large. Pourtant, NetBeans le possède lui aussi.

QUELQUES PROJETS EN COURS OU À VENIR

Projet Panama

Big data + machine learning
Modification dans JNI
Accès bas niveau (matériel) depuis un code Java
L'idée est de mieux connecter la JVM et les API non-Java (typiquement le code natif)

Project Valhalla

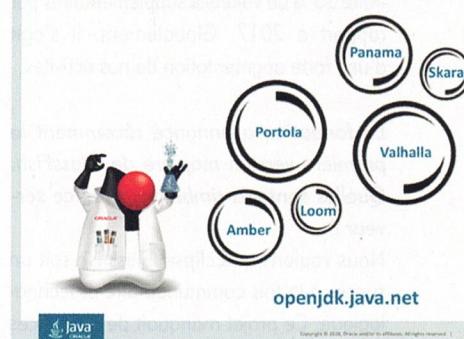
Optimisations au cœur du langage et de la JVM : value types, nouveaux generics.
Projet en cours depuis 2014 et partiellement implémenté dans JDK 10.

Project Loom

modèle concurrent à forte montée en charge
Fibers : threads plus légers et simples à utiliser dans la JVM

Projet Amber

Ce projet doit introduire plusieurs évolutions du langage : amélioration d'enums et des lambda, notion de pattern, classes data



Project Amber

- Local-Variable Type Inference
- Enhanced Enums
- Lambda Leftovers
- Pattern matching
- Switch expression



Java EE à la Fondation Eclipse



Nous avons posé quelques questions sur les activités Java à Mike Milinkovich, directeur exécutif de la Fondation Eclipse.

La fondation Eclipse a récupéré Java EE et TCK ainsi que GlassFish. Un an après, où en êtes-vous ?

Les contributions Jakarta EE et Eclipse Glassfish d'Oracle ont créé 39 nouveaux projets, avec environ 160 nouveaux committers, ils représentent 15 à 20 % de toute l'activité de la communauté Eclipse ! Notre équipe gérant la propriété intellectuelle a traité 30 % de volumes supplémentaires par rapport à 2017. Globalement, il s'agit d'une forte augmentation de nos activités.

La fondation a annoncé récemment la première version majeure de GlassFish. Quelles sont vos ambitions avec ce serveur ?

Nous voulons qu'Eclipse Glassfish soit un succès à la fois communautaire et technologique. Ce projet manquait de ressources dédiées ces dernières années au sein d'Oracle. Nous espérons voir une augmentation du nombre de contributions maintenant que la communauté est aux commandes. Nous espérons ainsi qu'il deviendra une référence sur les runtimes Java.

Les TCKs Java EE sont désormais open source, pourquoi est-ce si important ?

C'est vraiment une annonce importante. Depuis la création de Java EE, les TCKs ont toujours été très confidentiels. Cela a clairement freiné l'innovation puisque toute personne intéressée souhaitant développer

des implémentations « compatibles » devait passer un accord juridique très complexe et confidentiel avec Sun Microsystems (puis Oracle). Suite au passage des TCKs en open source, nous nous attendons à voir une augmentation du nombre d'entreprises souhaitant investir sur des outils et plateformes basées sur Jakarta EE. Cela ramènera de l'innovation ainsi que de la concurrence dans l'écosystème Jakarta EE.

Quel modèle de gouvernance avez-vous mis en place pour gérer et faire évoluer Java EE ? Quelles différences avec l'approche d'Oracle ?

Nous sommes en pleine phase de démarrage du processus de spécification. La différence la plus importante sera que celui-ci sera réellement indépendant, et attaché à aucun fournisseur de solution commerciale. Ce que vous remarquerez c'est que le super « spec lead » du JCP n'existera plus ! Les spécifications évolueront à l'avenir de manière collaborative, dans un processus plus proche des pratiques open source.

Qui supportera les versions actuelles et passées de Java EE ?

Sur ce point, il n'y a pas de changement. Des sociétés telles que IBM, Red Hat, Tomitribe ou encore Oracle continueront à fournir un support commercial pour leurs environnements et outils Java EE pour les années à venir.

Quelles sont les différences entre EE4J et Jakarta EE ?

A moins que vous ne soyez un committer sur un des projets de la fondation Eclipse directement liés à Glassfish et Jakarta EE, vous n'avez pas besoin de connaître EE4J ! Eclipse Enterprise for Java (EE4J) est le métaprojet contenant les contributions d'Oracle. Jakarta EE est le groupe de travail

et la « marque » sous laquelle les nouvelles spécifications sont publiées.

Pourquoi Java est-il si important pour le Cloud ?

Java est déjà l'un des langages et plateformes les plus importants dans le Cloud. Il est pris en charge par AWS, Azure et Google. C'est important car les entreprises ont déjà de nombreux développeurs Java expérimentés, maîtrisant le langage et les APIs, et sachant développer des applications critiques et scalables. Le défi pour Jakarta EE consiste à apporter les modifications nécessaires afin d'utiliser efficacement les nouvelles technologies de déploiement telles que Kubernetes et Docker, et à fournir des standards indépendants des fournisseurs pour écrire des micro-services en Java.

Microservice, infra-as-code, serverless, conteneurs... se sont des sujets chauds du moment. Comment Jakarta EE et GlassFish vont-ils supporter ces nouveaux modèles ?

Eclipse MicroProfile montre déjà la voie. Payara et d'autres proposent déjà des solutions open source basées sur Glassfish et MicroProfile, qui démontrent comment ces technologies peuvent fonctionner ensemble.

Les communautés Java étaient très préoccupées par le silence d'Oracle sur l'avenir de Java EE. Comment travaillez-vous avec la communauté et quelle sera la roadmap 2018-2020 ?

Des annonces seront faites à Oracle CodeOne (du 22 au 25 octobre) et à EclipseCon Europe (du 23 au 25 octobre)⁽¹⁾. Stay tuned!

(1) Le magazine a été imprimé au moment de ces conférences. Plus d'informations sur www.programmez.com