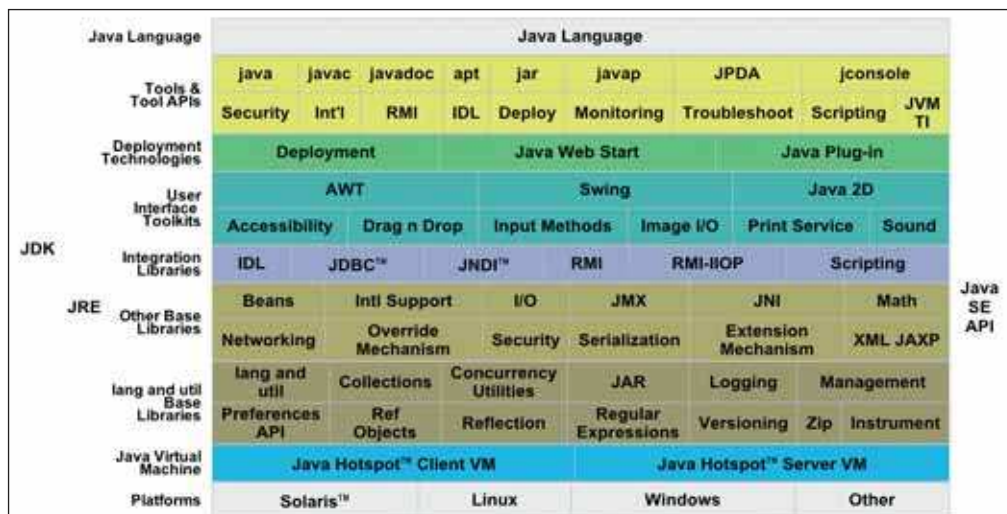




## 2008-2009 : Où en est Java ?

*Le monde Java continue à évoluer, malgré une activité relativement calme et une communication assez limitée depuis quelques mois. 2008 - 2009 annoncent pourtant de grands chamboulements dignes d'un Java 5 ou d'un Java EE 5.*



ressenti depuis quelques semaines, Sun n'organisera pas de Sun Techdays en France comme ce fut le cas en 2007. L'éditeur souhaite toutefois être présent sur différents événements comme dernièrement avec Paris on Rails et le salon Solutions Linux. Cependant, l'éditeur réfléchit à organiser une journée spécifique Java, la JavaDay qui avait eu lieu il y a quelques années. En cas de tenue, elle aurait lieu courant juin. Reste à confirmer la conférence et à définir le contenu et surtout les intervenants. Nous en saurons plus courant février ou mars.

### Java 6 Update N : mi-2008

Actuellement en développement, Java 6 Update N introduira un certain nombre de nouveautés importantes pour les applications et développeurs Java. La version finale devrait arriver aux alentours de JavaOne (mai prochain), aucune date précise pour le moment. Cette mise à jour est plus connue sous le nom de code : Consumer JRE. Le Consumer JRE fait partie intégrante de l'Update N et doit apporter à Java une modularité, des performances au démarrage

et en déploiement, jamais vues. Pour faire simple, Update N se concentre sur :

- un nouveau déploiement avec notamment un nouveau toolkit,
- l'apparition du Java Kernel,
- une nouvelle version du Java Plug-In.

### Consumer JRE

Le Consumer JRE doit fournir un environnement runtime Java (ou JRE), plus petit, plus modulaire, avec un temps de démarrage réduit, une installation simplifiée, des performances graphiques à la hausse (avec une meilleure intégration de JRE sur le système hôte). Pour cela, il inclut le Java Technology Deployment Toolkit, une suite d'outils et de fonctions permettant de simplifier la détection du JRE et de son installation. Surtout, avec le Consumer JRE, le JRE devient (enfin) modulaire. Modulaire, dans le sens, que le JRE installe uniquement ce que l'utilisateur a besoin pour faire fonctionner ses applications Java. C'est-à-dire que l'on télécharge (en transparence pour l'utilisateur) uniquement les modules dont l'application a besoin. Si par exemple, une nouvelle application a besoin de fonctions non instal-

lées dans le JRE, celui-ci va télécharger et installer les fonctions manquantes. Cela implique que le temps de démarrage sera rallongé la première fois, le temps d'installer. En procédant ainsi, on évite de télécharger un JRE de plus de 10 Mo ! Toujours sur le temps de démarrage, Consumer JRE doit fournir un processus de lancement plus rapide avec l'utilisation d'un cache disque en y chargeant des portions du JRE, ce qui doit réduire le temps de lancement d'une application.

### La fin de l'enfer des classpath avec Java Module

Un des problèmes récurrents de Java concerne le classpath et le fameux "classpath hell". Cela intervient lorsque l'on a différents JRE / JDK installés, plusieurs versions de la même librairie, des mêmes classes, différents IDE Java, etc. Bref, comment gérer les versions ? Les développeurs Windows étaient habitués à gérer tant bien que mal le fameux "hell dll". Dans le monde Java, on parle souvent de *classpath hell*, de *jar hell* et de *Extension hell*. Pour remettre un peu d'ordre dans le déploiement et la gestion interne, on disposera de Java Module, qui

introduit le concept du "super jar". Il doit à la fois simplifier le développement et le processus de déploiement. Pour reprendre la formulation Sun (JSR 277), il s'agit de "copier" les principes de fonctionnement de Maven.

À quoi va servir Java Module ? L'ambition est de faire du versioning, c'est-à-dire de gérer les versions des classes, librairies, des JAR, et d'éviter des conflits entre différentes versions, installées dans différents répertoires. Il intègre une gestion de dépendance (de quelle classe et librairie ai-je besoin pour mon application), le tout reposant sur un référentiel local (avec sa base de données). C'est donc une avancée non négligeable pour les développeurs Java. Car aujourd'hui, le JAR ne gère pas le versioning.

Pour ce faire, Java Module introduit les éléments suivants :

- JAM : nouveau format de Java Module, basé sur le format Jar. Il inclut en plus du Jar normal la signature du fichier jam (pour identifier, tracer le fichier), la compression des fichiers jam via Pack200, qualifiée d'hypercompression (JSR 200). Java 6 n'est que la première étape, Java 7 apportera de nouvelles fonctions

# Événements

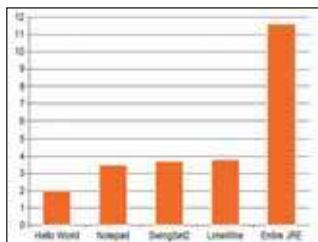


Jam, notamment sur la sécurité.

- Repositories : le référentiel est un " Maven like ". On dispose par exemple d'une hiérarchie.
- Support runtime : nouveau mécanisme durant le démarrage des applications, toutes les fonctions seront exécutées par la machine virtuelle et non dans un code tiers. Le runtime aura par exemple la validation des modules permettant de mieux contrôler les ressources et les modules réellement utilisés par l'application.

Sur Java Plug-In, l'implémentation a été entièrement revue. Le développeur pourra mieux spécifier un JRE spécifique pour exécuter une applet et les applets signées sont supportées dans une exécution Vista. Il est compatible avec IE 6 et 7 (XP et Vista). Il supporte Firefox 3 mais pas Firefox 2 et aucun support officiel n'est prévu selon les documents officiels. L'ancien Java Plug-in (ou Classic Java Plug-in) est toujours présent et on peut basculer d'un modèle à un autre.

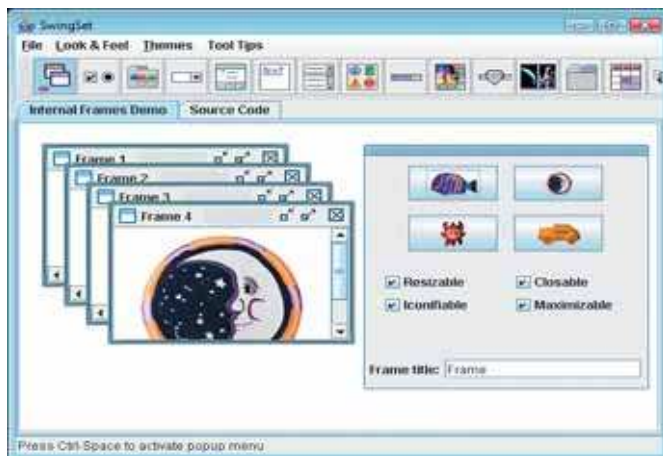
## Java Kernel : au cœur du futur de Java



Comme vu ci-dessus, le Consumer JRE a pour but d'être modulaire et de permettre un déploiement adapté du JRE. Le cœur de cette idée s'appelle Java Kernel. L'idée, vous l'aurez compris, est de proposer un JRE minimum avec le code et les fonctions pour fonctionner et nécessaires pour faire fonctionner la JVM et les applications. Jusqu'à présent, le JRE se composait de centaines de bundles se téléchargeant un par un. Le but est donc de proposer un " core java " rassemblant ces bundles de fonctionnement et qui soit le plus petit possible,



SE 6 == VICTORY!



Les nouveautés Swing

entre 2 et 3 Mo. Et au fur et à mesure des besoins, le JRE téléchargera en arrière-plan les bundles nécessaires.

## Du nouveau chez Swing

La partie graphique, souvent présentée comme une faiblesse, connaît une mise à jour avec Java 6 Update N. Cela aura un impact conséquent quant à l'exécution Java sur Windows avec une réécriture du pipeline graphique afin de prendre en compte Direct3D, ce qui permettra d'améliorer les rendus, et de mieux supporter la transparence. Même Java 2D pourra profiter de ce support et de l'accélération graphique matérielle. Concernant l'apparence, on bénéficie d'un nouveau Look and Feel nommé Nimbus, qui est une mise à jour des thèmes Ocean et Metal. Cette mise à jour permet de mettre à niveau les thèmes avec l'apparence des systèmes Vista et MacOS X 10.5 (et les nouveaux Linux).

## Java 7 : 2009

À l'heure où nous écrivons, le JSR lié à Java 7 n'était pas encore créé, signifiant que la liste des

fonctions propres à Java 7 demeure ouverte et susceptible de bouger. Cependant, comme nous l'a précisé Alexis Moussine-Pouchkine (Sun France), l'expérience tirée de Java 5 permet de savoir ce qu'il faut faire et ne pas faire quand on modifie en profondeur l'architecture et les fonctions du langage. " Si les annotations ont été un succès, ce n'a pas été le cas des " generics " indique Alexis. Car si les développeurs utilisent volontiers les generics, les écrire reste souvent une corvée. Il existe d'ailleurs un débat autour des modifications langage à faire, notamment sur les " closure ". Une des " craintes " étant d'apparaître comme suiveur de C#... Java 7 est comparable, en termes de nouveautés, de modifications, à Java 5. Par comparaison, Java 6 se veut dans la continuité de Java 5, même si la Upgrade N comporte des rajouts importants (mais pas au niveau langage). Et si Java 6 reste un projet propriétaire, Java 7 sera la première vraie version open source (via le projet OpenJDK) de l'histoire de Java ! Côté agenda, Java 7 sera disponible courant 2009,

peut-être durant le 1er semestre, soit un peu plus de 2 ans après la sortie de Java 5. Si Java 5 a été bien adoptée, malgré une lente percée à cause du manque de support dans les outils de développement, Java 6 l'a été par environ 20 % des développeurs (dixit Alexis). Sun s'attend d'ailleurs à des migrations directes Java 5 vers Java 7, qui reprendra les bases de Consumer JRE pour les étendre. De nombreuses nouveautés et améliorations sont attendues dans Swing (Swing Application Framework notamment), évolution des génériques, annotations étendues, disponibilité de NIO 2 (JSR 203), amélioration du support XML, support du javabeen property, des closures, de JMX 2, etc.

## Java EE 6 : courant 2009

Sur le prochain Java EE, la v6, les choses évoluent elles aussi. Sur la v5, cela se déroule plutôt bien, même si on attend l'implémentation JBoss et Websphere (hors version community). Java EE 6 apporte les nouvelles versions suivantes : Servlet 3.0, JPA 2 (désormais disponible séparément des EJB), EJB 3.1, amélioration autour de JCA. La disponibilité est attendue pour le 1er trimestre 2009. Et l'implémentation de référence sera disponible en même temps (Glassfish v3).

## JavaFX : trop immature

Présenté en fanfare à la JavaOne 2007, JavaFX était présenté comme une alternative aux plates-formes Adobe et Microsoft. Depuis, la technologie souffre de son immaturité et on en parle très peu. De l'aveu même de Sun, il n'y a pas grand-chose de nouveau depuis le printemps dernier. Mais les développeurs travaillent beaucoup à améliorer la plate-forme, le langage de script. La disponibilité prochaine de Consumer JRE aidera sans aucun doute JavaFX.

■ François Tonic