

Evolution de Java

Les développeurs

Evolution des APIs

Plateformes

Adoption de Java

La vision des “autres”



JavaOne Conference 1998

14000 participants

Exposés:

- Java Technology in Embedded Devices (3)
- Core JDK Technology Update (4)
- Java Commerce and Card Technologies (4)
- Tips and Techniques to Help Developers (6)
- Java Media APIs (7)
- Industry Momentum (11)
- New APIs and Core JDK Classes (13)
- Enterprise APIs and Technologies (21)
- Java Technology in the Real World (25)

KNUCKLETOP: THE JAVA RING



“Bague à puce” avec une Java machine

Applets téléchargeables:

- se connecter aux machines du Hackers' Lab,
- calculer une partie d'une image géante
- obtenir une tasse de son café favori au distributeur



Evolution des API

API centrale: 1.0, 1.1, 1.2 (bêta)

Définie de facto par le JDK de Sun

Incorporation d'APIs externes dans l'API centrale:

- JDBC (entrée dans 1.1)
- RMI (entrée dans 1.1)
- JFC (entrée dans 1.2) (Amélioration de l'interface, Collections, etc.)

Développements d'APIs pour les standards:

- API pour les bases de données ODBC => JDBC
- API pour QuickTimeTM (Apple)



Nouveaux APIs

InfoBus. Echange de données structurées entre Beans.

JavaBeans Activation Framework (JAF). Pour rendre un Bean capable de traiter des données venant d'autres sources.

Java Card. Pour la programmation des cartes à puces.

Java Mail. Modèle de messagerie indépendantes de la plateforme et du protocole.

Java Management. Gestion de réseaux hétérogènes.

Java Media and Communication. Utilisation de medias interactifs: son, animation, 2D, 3D, téléphonie, parole, collaboration.

EmbeddedJava. API pour les systèmes embarqués

PersonalJava. API pour appareils domestiques.



Enterprise JavaBeans

Extension du modèle des Beans aux composants serveurs.

Architecture multi-rangs (multi-tier)

- clients fins
- multiples serveurs

Les composants serveurs s'exécutent à l'intérieur d'un "component execution system"

Définit les ressources que doit offrir le système d'exécution: transactions, processus, etc.



Multiplication des machines virtuelles

Une machine réelle --->

N machines virtuelles + bibliothèques de classes:
(p.ex. Netscape, Explorer, JDK, Symantec Café)

Solution Apple: Macintosh Runtime for Java

- une MV et un bibliothèque pour toutes les applications

Solution Sun: JumpStart

- un CD-ROM contient un environnement Java standard pour chaque plateforme (PC, Unix, Mac, etc.)
- télécharge l'environnement sur chaque machine

==> Découpler l'évolution des applications et des OS de celle de Java.



La plateforme Java de Sun

4 implementations:

JDK API. Pour PCs, stations de travail, et serveurs

PersonalJava™ API. Pour appareils “ménagers”:
téléphones, TV, jeux, ...

EmbeddedJava™ API. Pour appareils avec contrôleurs:
automobiles, usines intégrées, ...

Java™ Card™ platform . Pour cartes et anneaux à puces

Nombre de téléchargements du JDK: 2'500'000.



JavaPC

- Convertir un “vieux” PC en Network Computer
- Installation de JDK 1.1 partout
- Fonctionne avec DOS ou Windows 3.x
- Les applications DOS/Windows restent disponibles
- Migration vers le NC.



Adoption de Java [BYTE mars 1998]

Etude de 269 lecteurs de BYTE qui ont évalué Java

- développeurs, cadres supérieurs, manager techno.
- taille moyenne des organisations: 620 employés

Utilisation de Java

43% l'utilisent déjà, 31% bientôt, 26% ne l'utiliseront pas

Pourquoi ne pas utiliser Java

29% trop tôt, 17% trop compliqué, 12% trop cher, 10% mauvaises performances, 9% trop lent

Pourquoi utiliser Java

Utilisation sur > 1 plateforme: 80% des réponses

Accès aux bases de données: 79% des réponses



Etude (suite)

Vitesse d'exécution (pour ceux qui l'utilise-ro-nt)

3% plus rapide que les autres alternatives

53% suffisamment rapide pour la plupart des choses

38% lent mais acceptable

6% trop lent

Importance du programme de développement Java

45% non essentiel mais utile

35% "mission critical" et non essentiel

20% "mission critical"



Une autre vision du futur (sans Java)

d'après un article de Monsieur Bill Gates [Byte, mars 1998]

Trois facteurs ont dirigé l'évolution du développement du logiciel:

- adoption de standards
- programmabilité des fonctionnalités offertes aux utilisateurs
- extension de la portée de l'automatisation des applications et des informations dérivées (Internet)

Les deux développements les plus importants de ces dernières années:

1. la maturation d'un protocole d'application commun: COM.
2. le développement d'un langage commun orienté-composants: VBA.



Le futur

Acheter ou reprendre une application et l'adapter

=> critère d'évaluation : facilité d'adaptation

=> comment réaliser l'adaptabilité et l'intégrabilité ?

Convergence des modèles d'objets

même modélisation dans différentes applications
composants jugés sur leur ressemblance avec des
modèle déjà connus

Intégration internet

indépendance de localisation
outils de mise au point adaptés au réseau
solution: DCOM, Directory service



Représentation (autonome) des données

*développer des règles pour représenter les données
comme des ensembles de concepts reliés plutôt que
comme des structures liées à une application spécifique.*

méta-contenu

solution: XML

«Composantisation»

ne charger que le code nécessaire

solution: fusion de ActiveX et DLL

combinaison de composants

solution: VBA comme langage de composition/scriptage

Amélioration de l'implication des utilisateurs

implication des utilisateurs dans l'adaptation des
applications

solution: macro recording en VBA