Urbanisation et architecture des systèmes d'information

Plate forme pour le composant logiciel 2/2

JAVA, JEE et les EJB

David Eudeline
eudeline.david@free.fr

JAVA

- Qu'est ce que JAVA?
  - Un langage très jeune (1995)
  - Un langage de programmation orienté objet
  - Un ensemble d'API standards variés
  - Une architecture basée sur une Virtual Machine
  - Un ensemble d'outils inclus dans un kit de développement et de la documentation
    - www.javasoft.com
    - www.java.sun.com

JAVA

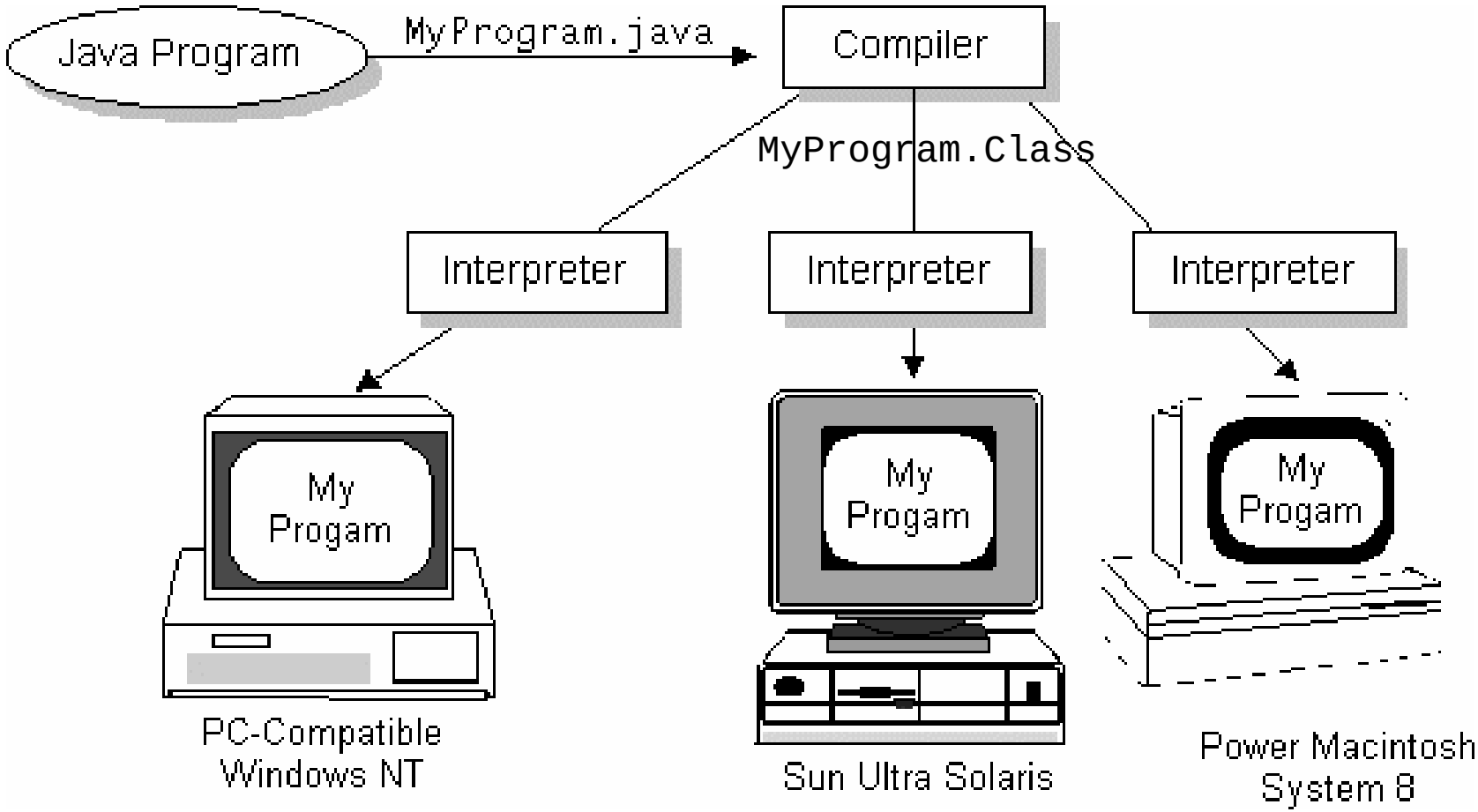
- Caractéristiques du langage
  - orienté objet
  - interprété
  - portable
  - simple
  - robuste
  - sécurisé
  - multi-threads
  - Distribué

  => pallie les faiblesses du C++, la programmation est facilitée.

JAVA

- JAVA a la particularité d'être en même temps compilé et interprété.
  - Avec un compilateur, le code JAVA est transformé en un langage intermédiaire indépendant de la plate-forme et formé de *bytecodes*.
  - Les instructions *bytecodes* sont ensuite interprétées puis exécutées par la machine locale au sein d'une JVM. La JVM n'est pas une machine à proprement parler mais un programme natif qui se charge de convertir tous les *bytecodes* en code binaire exécutable, une sorte d'émulateur.
  - Les *bytecodes* rendent possible la devise de JAVA : Write once, Run anywhere.
- Indépendance du code / plate forme
  - Code portable
  - Code mobile

JAVA


Java Program
MyProgram.java
Compiler
MyProgram.Class
Interpreter
Interpreter
Interpreter
My Progam
My Progam
My Progam
PC-Compatible
Windows NT
Sun Ultra Solaris
Power Macintosh
System 8

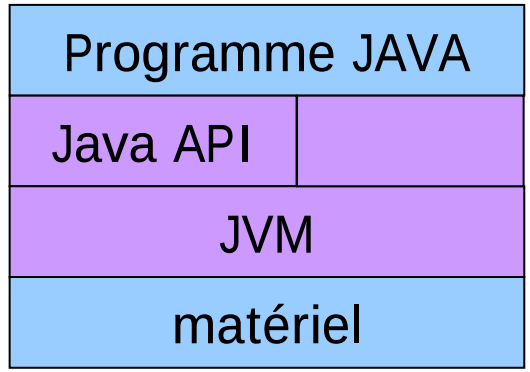
JAVA

Java est distribué

- API réseau simple mais efficace, notion de code mobile
- Chargement dynamique (applet)
- Servlet => Applet sur le serveur
- Remote Method Invocation(RMI)
- JavaIDL (CORBA)
- Les applets JAVA s'exécutent dans une *sand box* (bac à sable) et n'ont pas accès aux ressources physiques du système telle que la mémoire ou le système de fichiers

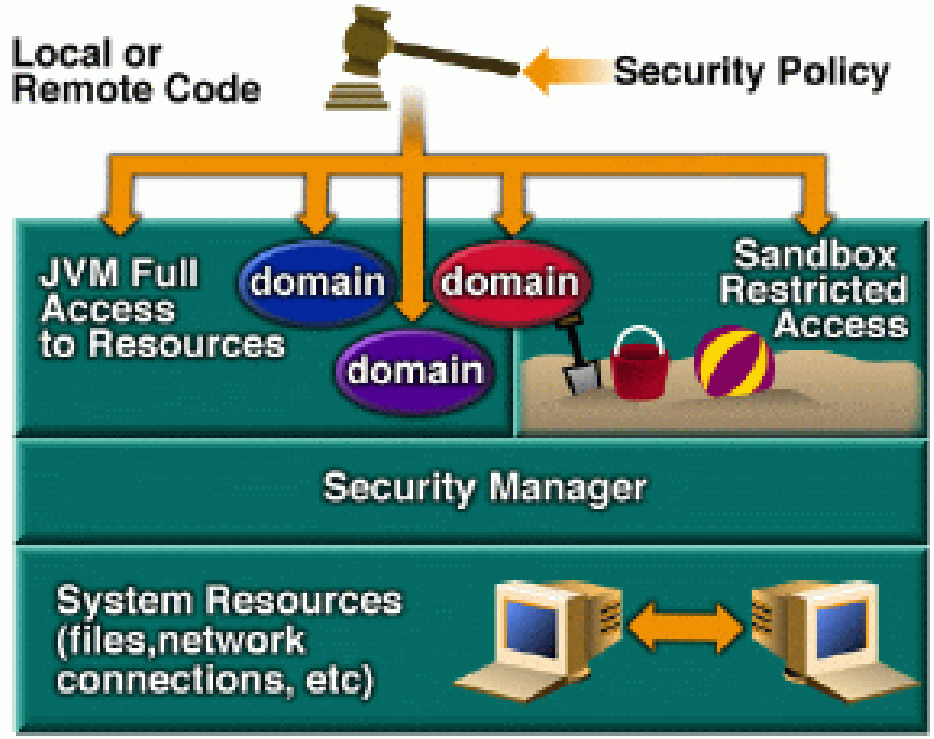
JAVA

- JDK (version actuelle 1.6)
  - Environnement minimum pour développer une application
    - compilateur, interpréteur (javac, java, etc.)
    - librairies standards
    - téléchargeable à partir du site SUN (http://java.sun.com)
    - détermine la richesse de l'API



  - Variables d'environnement
    - **CLASSPATH**: indique le chemin où se trouve les classes
    - **JAVA_HOME**: répertoire d'installation du JDK
    - **PATH**: doit contenir le répertoire du compilateur et de la machine virtuelle

JAVA: sécurité

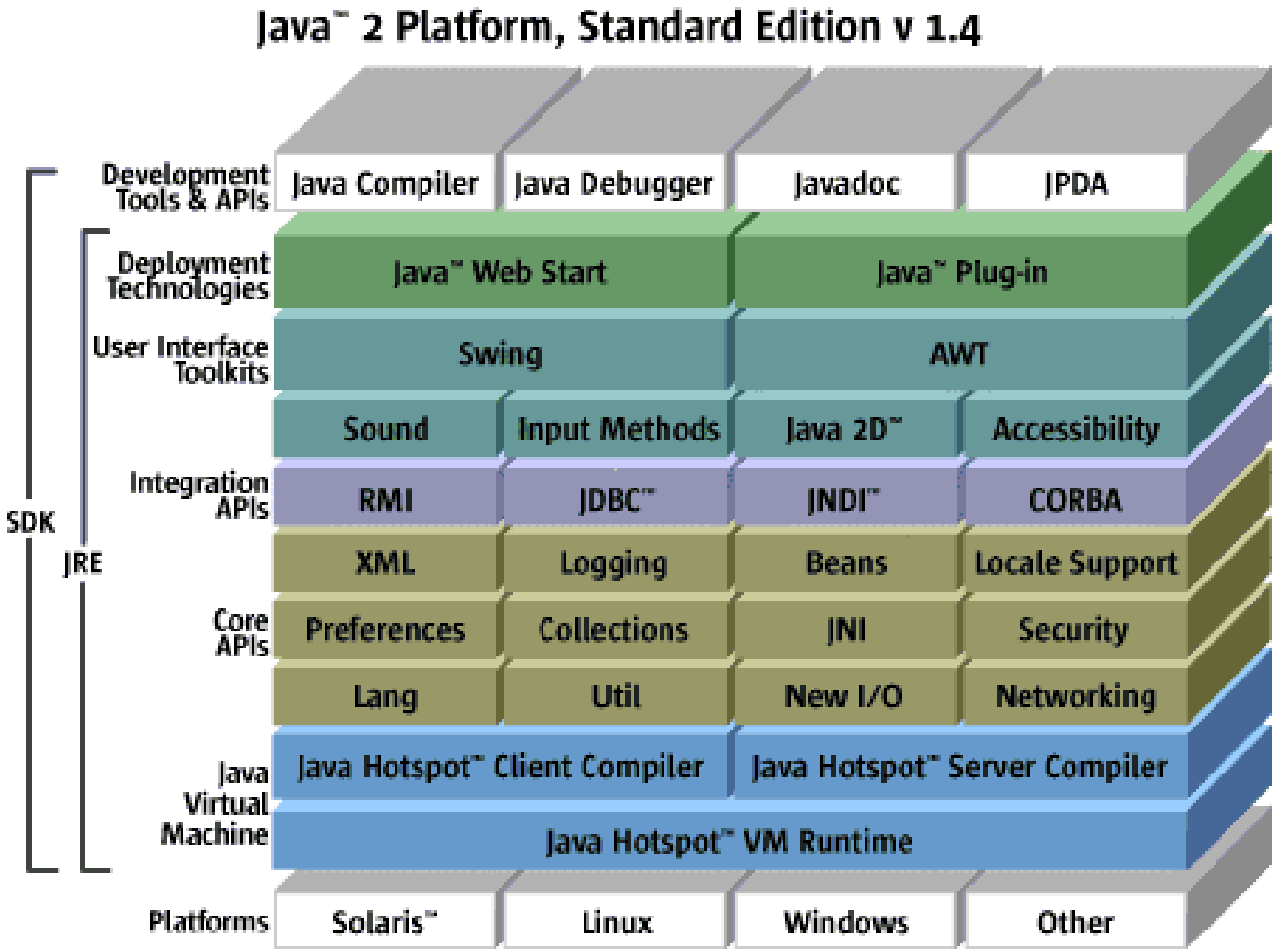


(Modèle de sécurité 1.2 -doc. Tutorial Java – Sun)

JAVA : API

API JAVA	Description
Composants essentiels	Objets, threads, strings, nombres, entrées/sorties, propriété du système, heure, date, …
Applets	Ensemble des conventions utilisées par les applets
Réseau	URL, sockets TCP et UDP, adresses IP
Internationalisation	aide pour écrire des programmes qui s'adaptent à des pay spécifiques et qui sont affichés dans le langage approprié
Sécurité	modules de sécurité haut niveau et bas niveau, incluant le signatures électroniques, le management de clés, le certificats et les contrôles d'accès
Base de données	fournit un accès à un large éventail de bases de donnée relationnelles

JAVA : Framework



Intégration des systèmes client/serveur

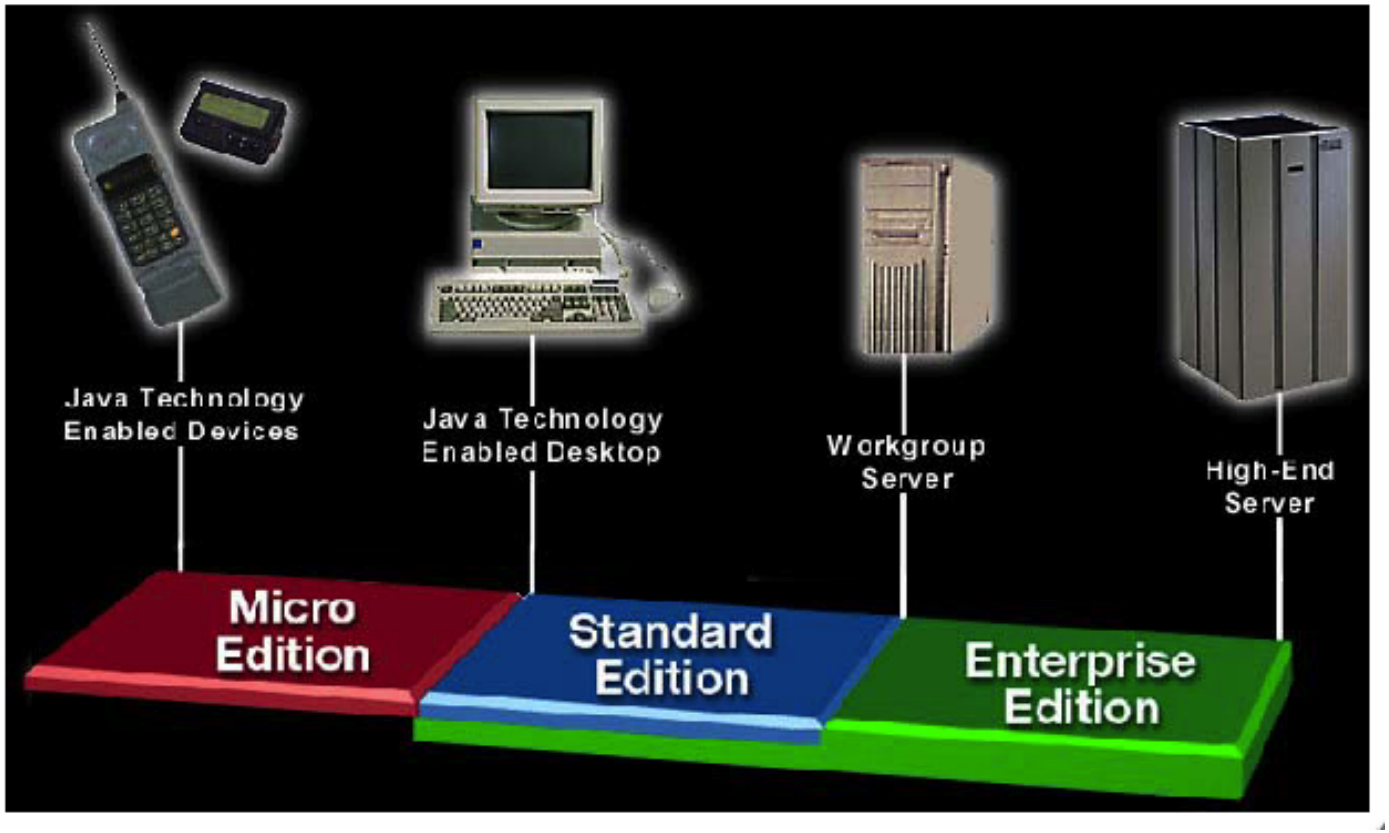
Plate forme pour les composants logiciels

JEE 5

JEE

- JEE (Java Enterprise Edition): Plate forme de développement d'applications multi-tiers
- JEE est un environnement intégré d'applications reposant sur:
  - Spécifications (API et plate forme)
  - Spécifications sur le comportement des serveurs d'applications.
  - Implémentation de référence des API gratuite et des produits du commerce
  - Acteurs: SUN, BEA Systems, IBM
  - Trois distributions: JEE, JSE, JME

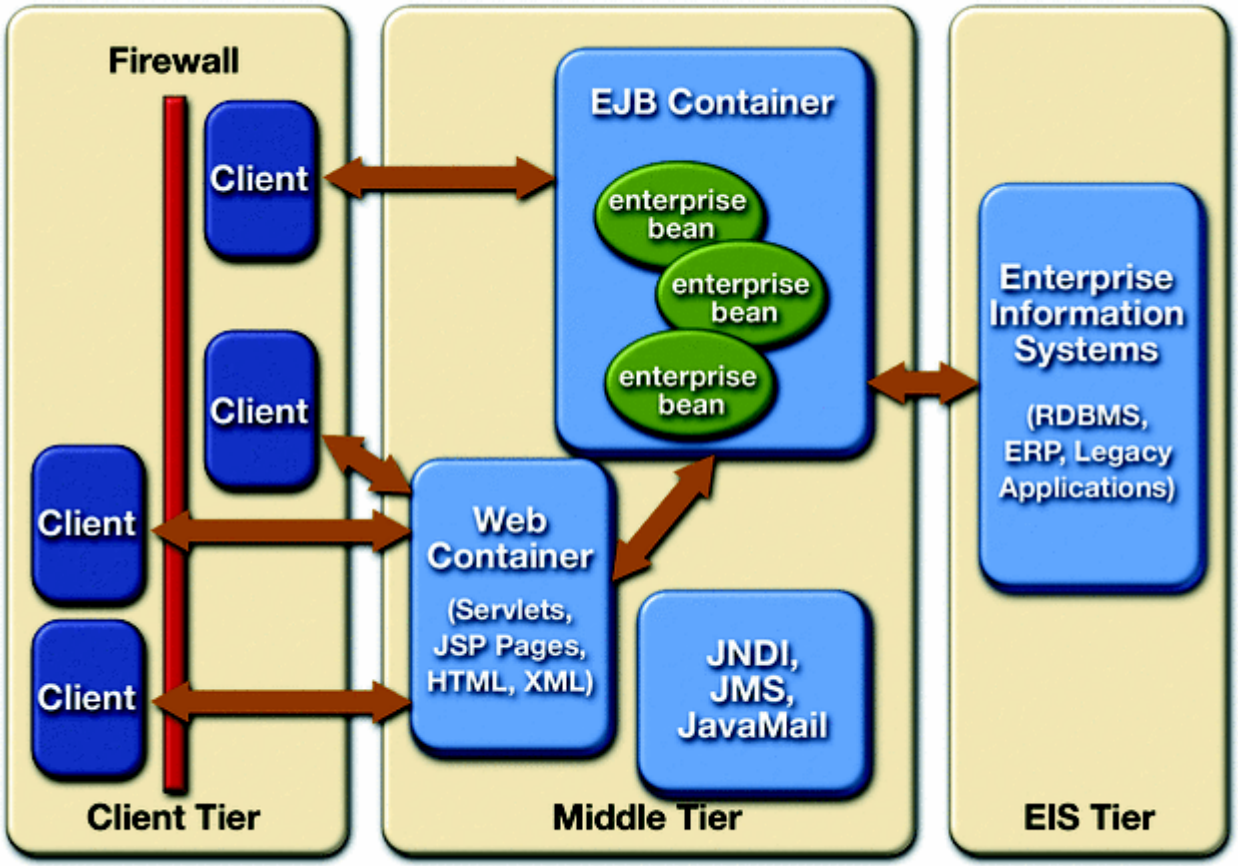
JEE = 3 distributions



JEE

- Les composants:
  - Les EJB (Enterprise JAVA Beans)
  - Les servlets (applets sur le serveurs WEB)
  - Les JSP (Java Server Pages)
  - Le serveur JEE fournit des conteneurs qui permettent de simplifier les composants et d'offrir tous les services nécessaires
  - Protocoles de communication (RMI)
  - Services

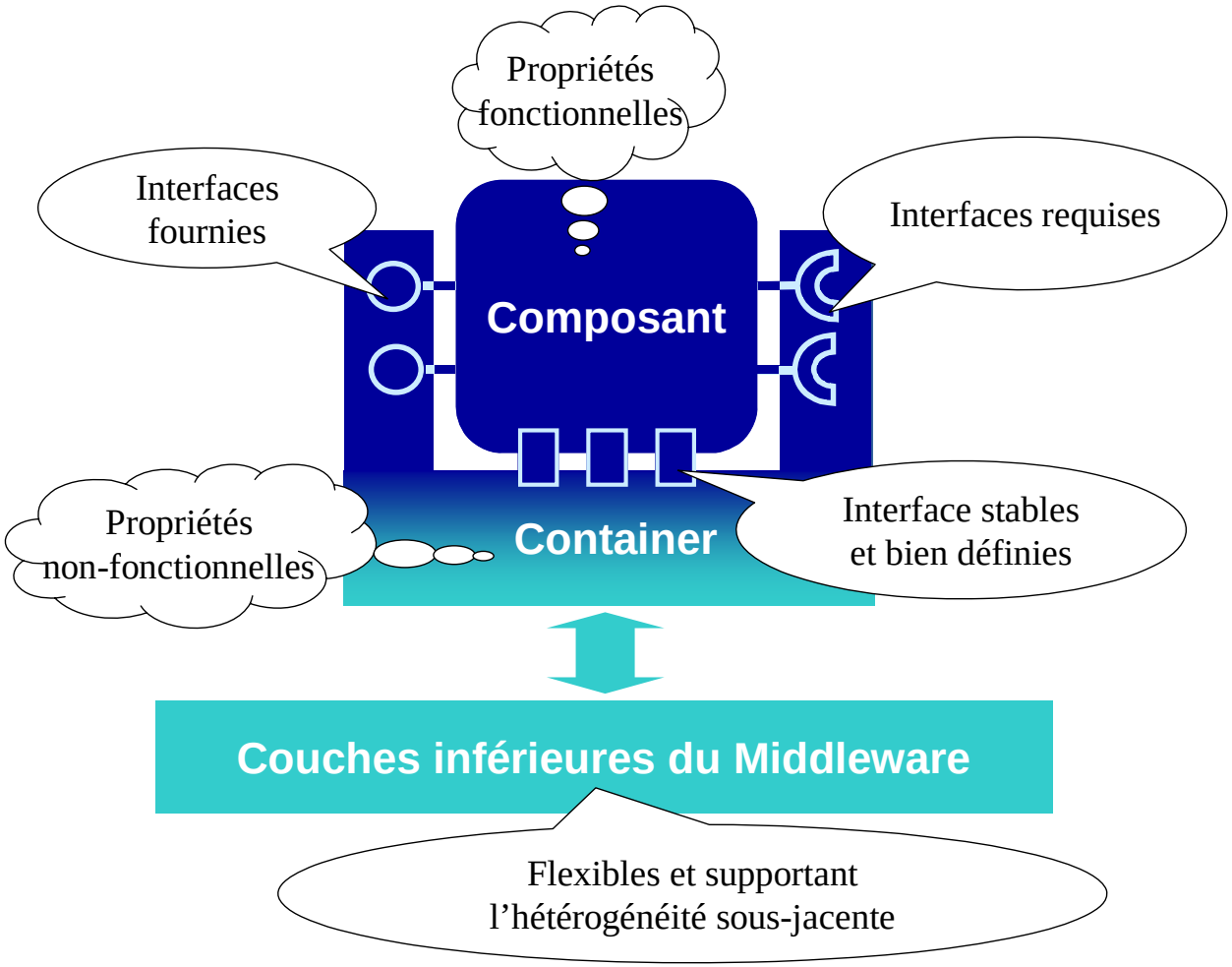
JEE: Architecture



JEE: Container

- « container »: fournit un environnement d'exécution pour les composants logiciels
- Services rendus:
  - transaction, sécurisation des accès, pooling des ressources, gestion de la mémoire, persistance des données, etc.
- Le serveur d'applications est le terme générique pour désigner un ou plusieurs containers fournis par un éditeur

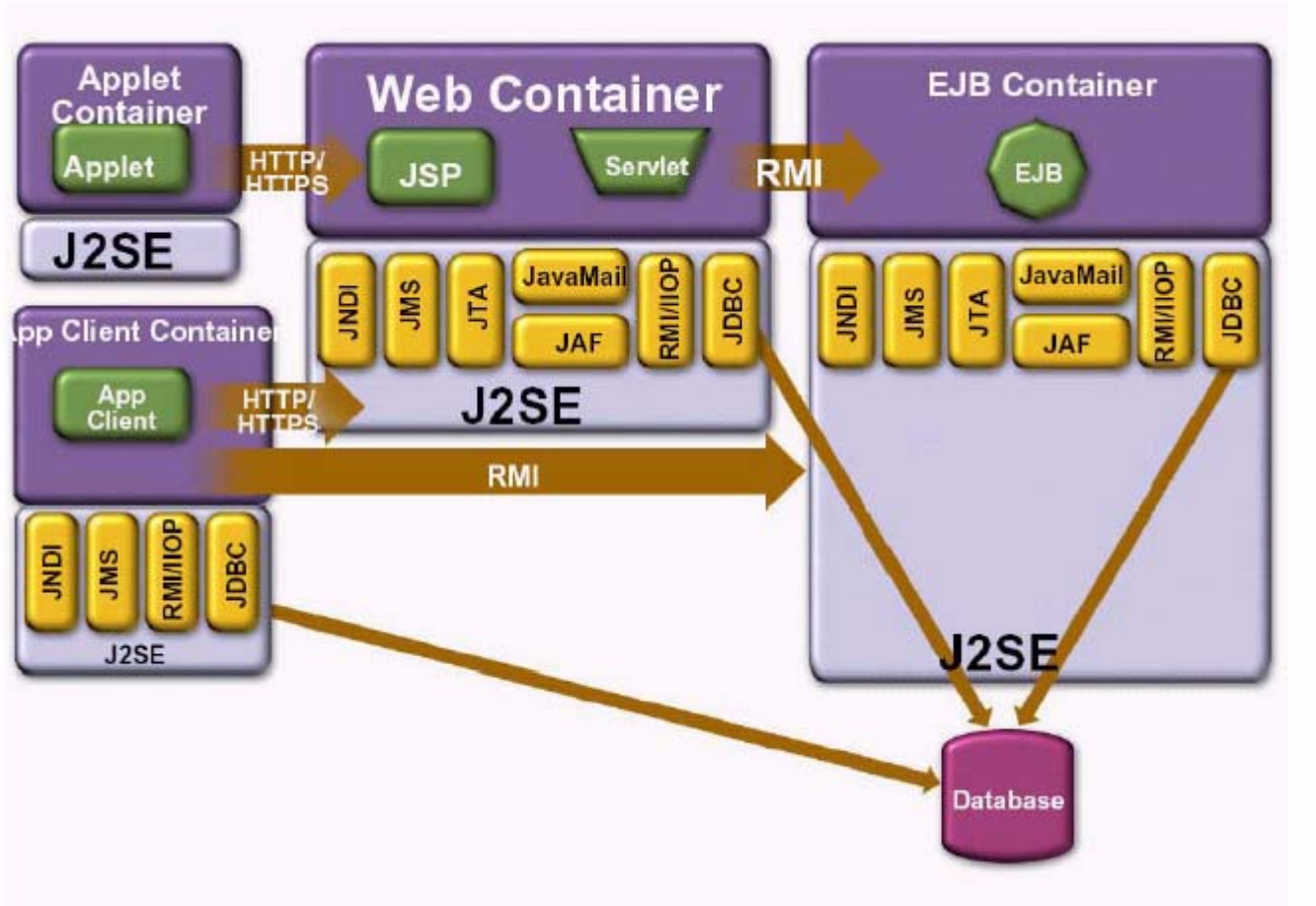
JEE: Container



JEE: Composants

- Trois types de composants
  - Les **composants web** : support pour la logique de présentation destinée à un navigateur (IE, netscape, ...). Ces composants s'exécutent dans le **container web** (Servlets, JSP)
  - Les **Enterprise Java Beans** (EJB) : support pour la logique métier et s'exécute dans le container du niveau métier encore appelé **container EJB**.
  - Le troisième destiné à enrichir une interface web est l'**applet** et s'exécute dans un container dit **container d'applet** résidant dans le navigateur web du poste client.

JEE: Composants



JEE: Composants WEB

- Les applications web proposent une IHM sous la forme d'une succession de pages HTML statiques ou dynamiques:
  - **Statique**: le contenu du document est fixe et stocké dans le système de fichiers du serveur web.
  - **Dynamique**: les pages sont générés par les composants web résidants dans le container web. Accès à la logique du tiers métier afin d'extraire les données
- JEE spécifie deux types de composants web : les servlets et les JSP (Java Server Page).

JEE: composants web

- Les servlets (applets sur le serveurs WEB)
  - prg autonome stockés dans un fichier .class sur le serveur
  - L'appel de l'URL déclenche l'exécution d'un moteur qui exécute le code compilé sous forme d'applet
  - Retour des données au format HTML
  - Persistance des données => Permet le suivi de l'activité du client et la gestion des formulaires

JEE: composants web

- Les servlets (applets sur le serveurs WEB)
  - Suivi de session
    - HTTP protocole non connecté
    - pour le serveur, 2 requêtes successives d'un même client sont indépendantes
    - Objectif : être capable de "suivre" l'activité du client sur plusieurs pages
  - Notion de session
    - les requêtes provenant d'un utilisateur sont associées à une même session

JEE: composants web

- Les JSP (Java Server Pages)
  - prg **source** Java embarqué dans une page **html**
  - Balise incluses dans la page html (<% et %>)
  - L'appel de l'URL déclenche l'exécution d'un moteur qui compile le code source JAVA contenu dans la page html exécute le code compilé
  - La JSP produit du code HTML en sortie d'applet interprétable par le navigateur
- JSP => utilisation frontale pour l'amélioration de l'ergonomie et couplage soit avec des servlets soit avec des EJB.

EJB: Enterprise Java Beans

- Technologie EJB pour la création de composants métiers transactionnels, sécurisés et distribués.
- Les EJB sont exécutés dans un container EJB qui offre des services:
  - cycle de vie, session, persistance, gestion de transaction, sécurité
- Il existe trois types :
  - **Entity Beans** ou EJB entité
  - **Session Beans** ou EJB session
  - **Message-driven Beans** ou EJB orienté message

JEE: EJB

- 4 technologies ont «inspiré» les EJBs
  - Java: pilosophie WORA : «Write Once, Run Anywhere»
  - JDBC: connecter facilement des applications à une BD
  - Servlet: étendre dynamiquement le comportement d'un serveur
  - Beans: «enficher» des entités autonomes sur une plate- forme
- **Spécifications EJB** définies par Sun
- **Contrat EJB**: Convention passé entre les acteurs du développement (conteneur, serveur, composants,etc.)
  - Développeur de composants, fournisseur de conteneur, fournisseur de serveur, installateur
- Implantées par Sun (J2EE) et d'autres constructeurs (IBM, Bull, ...)

JEE: EJB session

- composant orienté métier
  - Composant attaché à un seul client mais partageable
  - Ne contiennent pas de données persistantes
  - Il est détruit quand il n'est plus utilisé par le client
  - représente une fonction métier, les étapes d'une tâche
  - Lien avec le tiers données fait via JDBC ou un EJB entité
  - Sans état (***stateless***) ou avec état (***stateful***)

JEE: EJB session

- EJB Session avec Etat (***stateful***)
  - représente un processus métier qui s'effectue en plusieurs opérations (exemple : commande en ligne)
  - non partagé : dédié à un client déterminé
  - Utilisé en tant que « contrôleur »
- EJB Session sans Etat (***stateless***)
  - représente une fonction métier qui peut être effectué par une seule « méthode » (exemple : consulter son solde)
  - partageable entre clients, performants, peu exigeant (conteneur)
  - peut servir à garder en mémoire des informations stables exploitables par divers clients (exemple : informations de référence)

JEE : EJB entity

- Composant gérant des données
  - L'objet contient des données (hors du tiers données)
  - Mécanisme de persistance en écrivant les données dans la base
  - Partagé entre clients
  - instance créée à la demande d'un client
  - L'instance de l'EJB continue à vivre tant qu'il existe un client actif.
  - Lorsque l'instance est inutilisée, elle est passivée (Les données sont enregistrées dans la base de données) et peut être détruite par le container.
  - Gestion transparente pour le développeur.

JEE : EJB entity

- Gestion de la persistance
  - EJB entity CMP (container managed persistence)
    - composant entité géré par le container
    - utilise automatiquement les services de persistance fournis par le container
  - EJB entity BMP (bean managed persistence)
    - La persistance est activée par le container mais les mécanismes de persistance sont développés en spécifique
    - implémente sa propre méthode (codée par le développeur) pour assurer sa persistance
  - ⇒Outils complémentaire : JDO, mapping objet/relationnel (toplink de Oracle), XML)
  - ⇒EJB 3.0: Meilleure prise en compte de la persistance

JEE : EJB message

- Message-Driven EJB
- Les EJBs orientés message ont été introduits dans la spécification EJB 2.0.
- destinés à traiter les messages asynchrones reçus depuis une file d'attente respectant le protocole d'échange JMS (Java Messaging Service).
- Permet de déclencher un service de l'application de manière asynchrone.

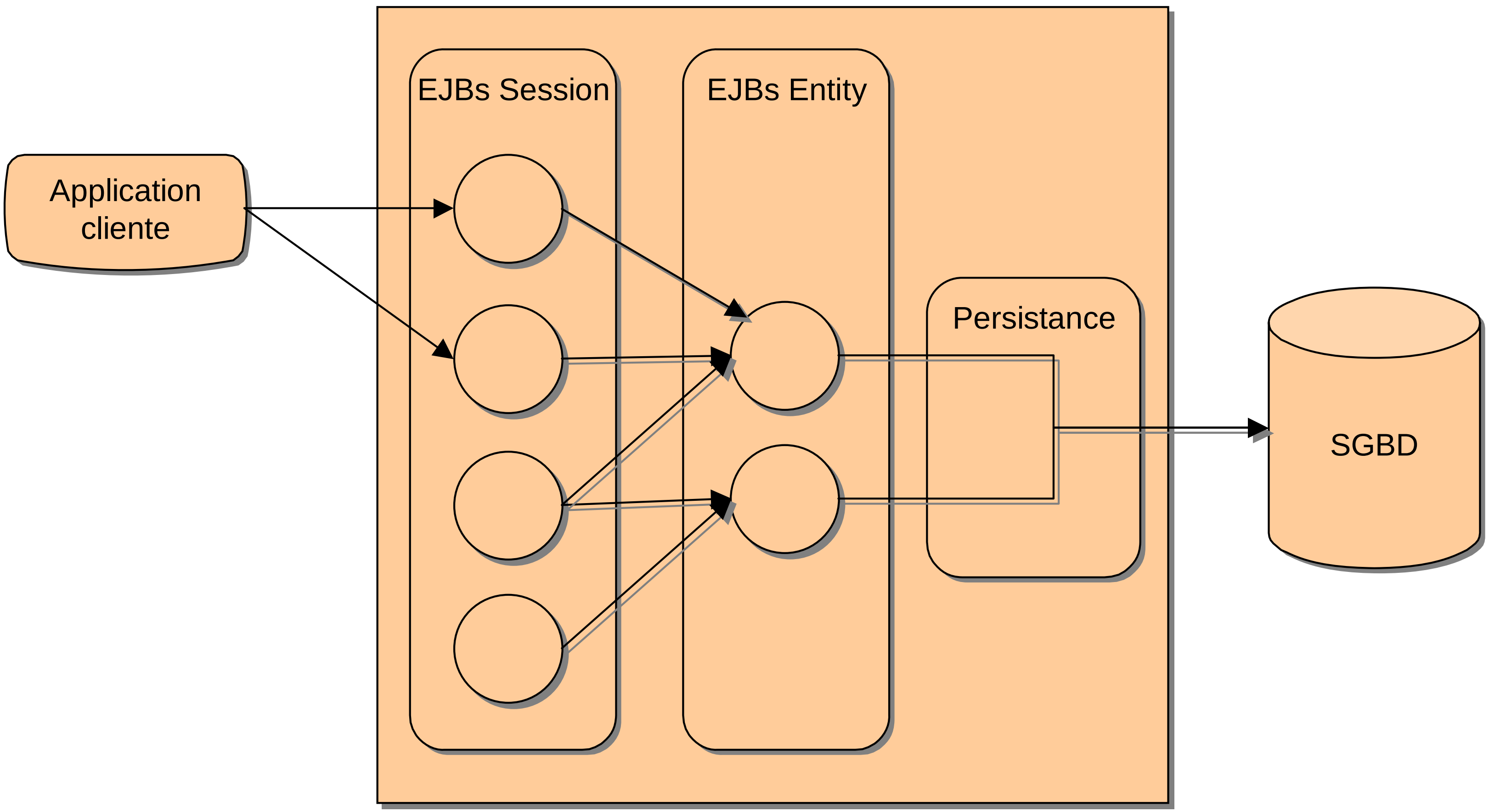
=> synchronisation des applications nomades.

J2EE: Les services

- Sécurité: Authentification/gestion des accès (X509 V3, login/mot de passe)
- Persistance (JDBC)
  - Accès aux base de données (Cf. ODBC) => Bean entité
- Transactionnel (JTS/JTA)
  - JTS: Service transactionnel fourni par les serveur d'EJB
  - JTA: API conforme à DTP (XA) pour la programmation des transactions
- Nommage (JNDI) => interop avec LDAP et DNS
- Asynchrone (JMS) => Gestion des MOM et des applications asynchrone

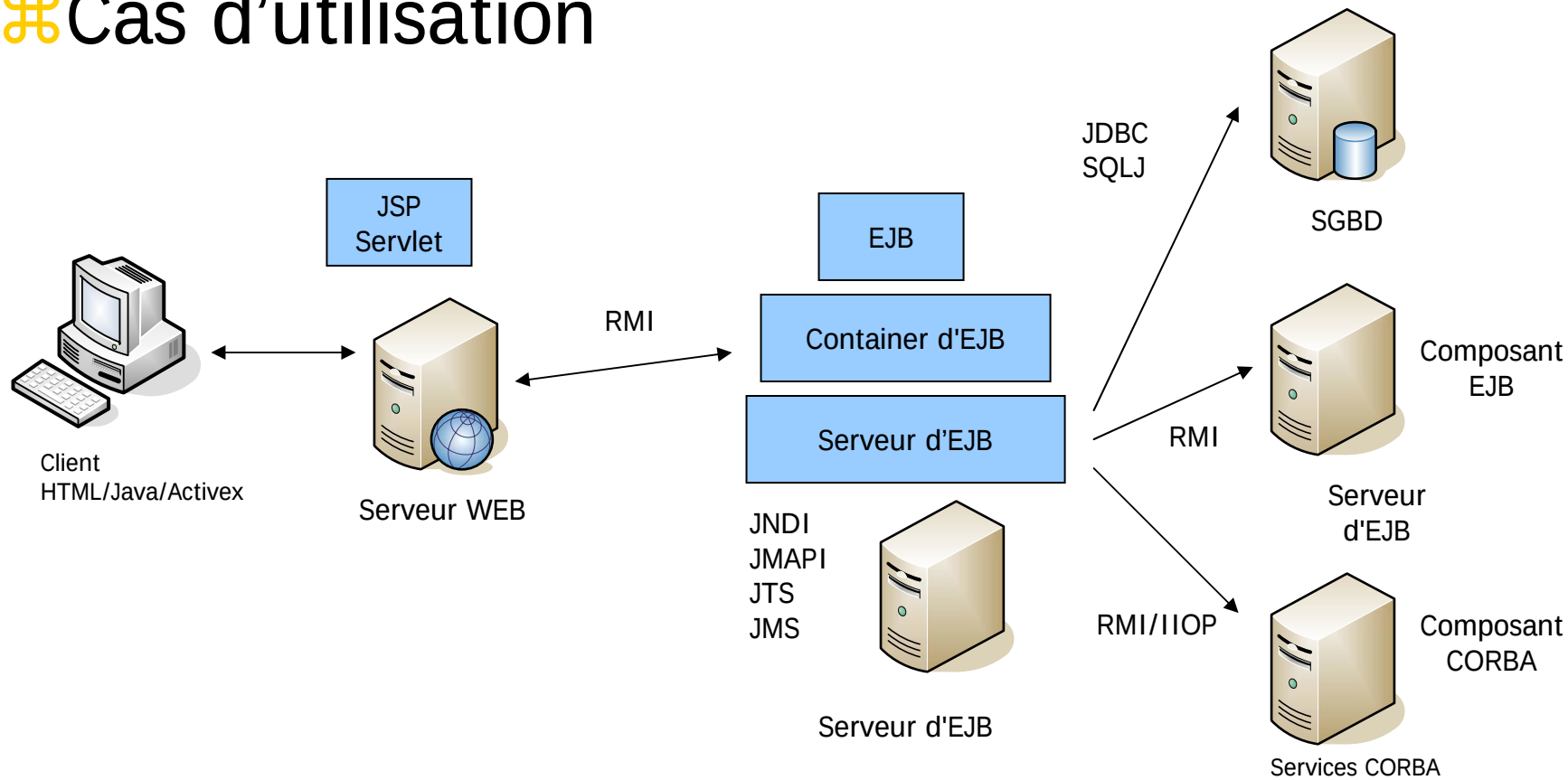
Les services de sécurité et de transaction sont fournis par les structures d'accueil

EJB / cas d'utilisation



JEE : EJB

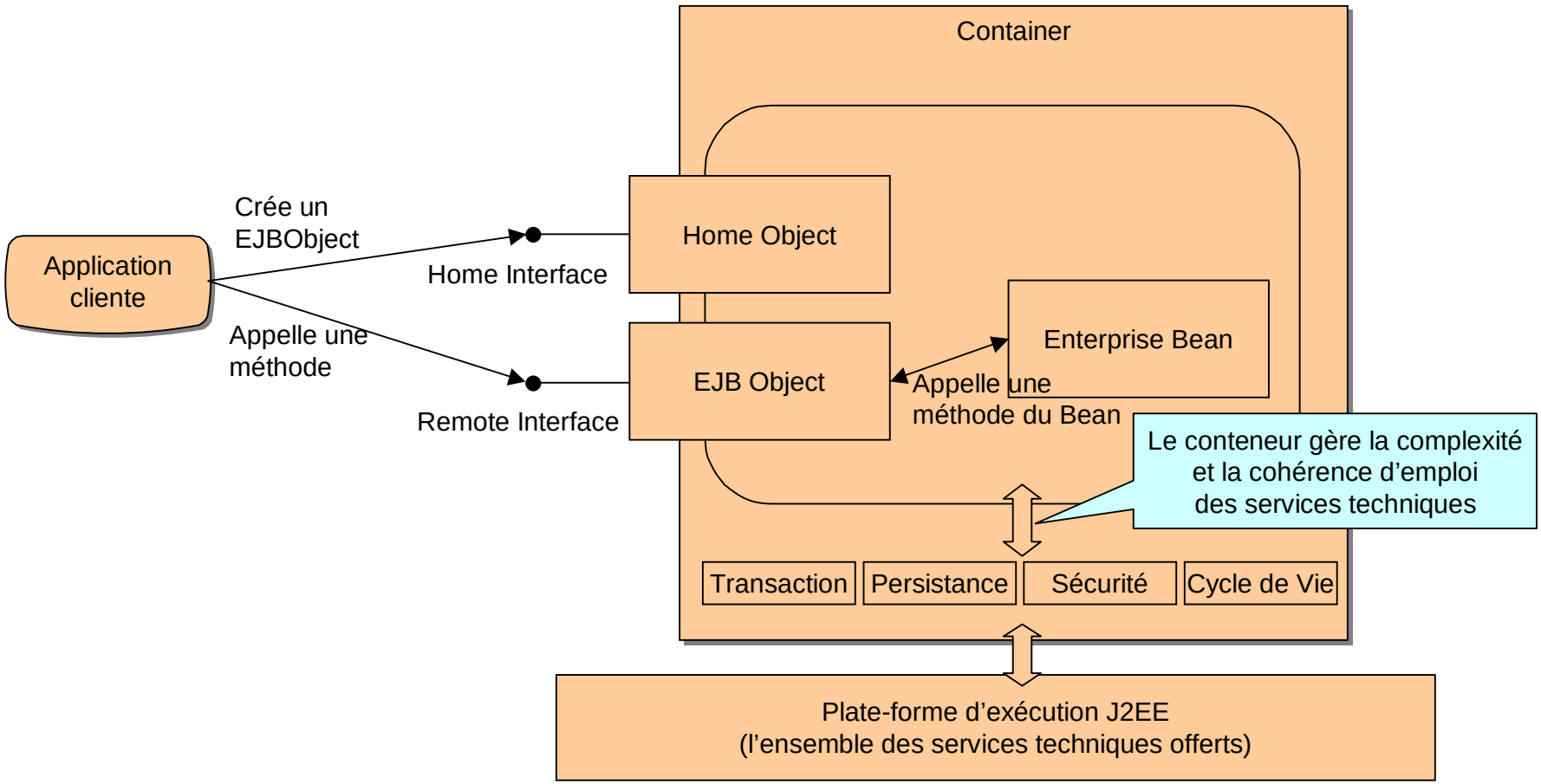
Cas d'utilisation



JEE : EJB

- Enterprise JavaBeans
  - Construction/devt/packaging

JEE: EJB

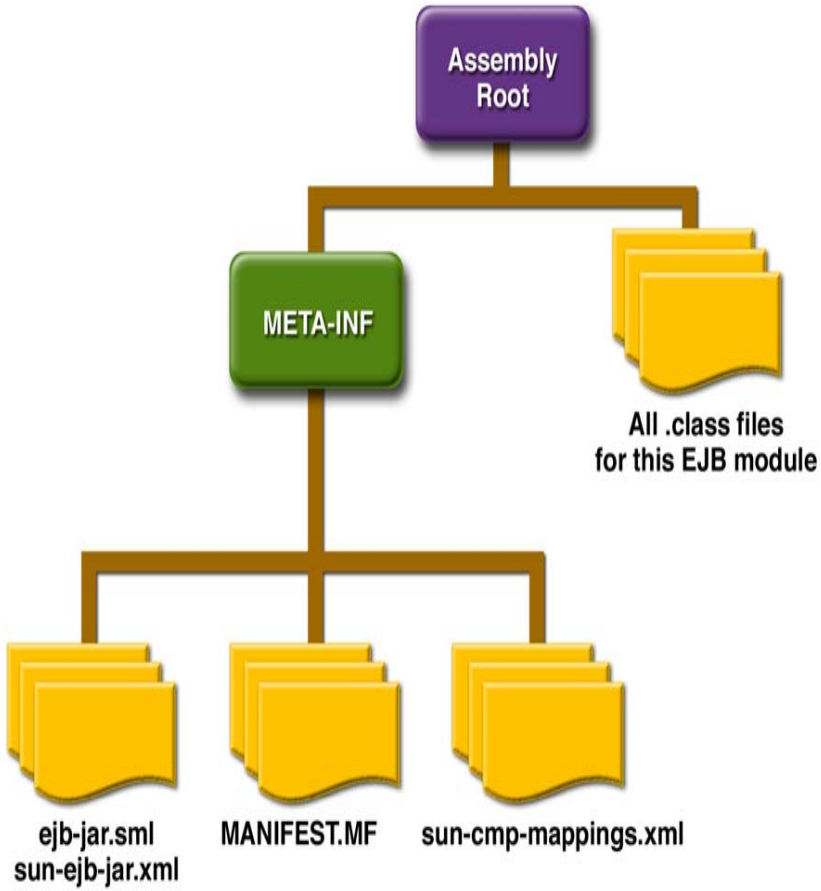

Container
Crée un
EJBObject
Application
cliente
Home Interface
Home Object
Appelle une
méthode
Remote Interface
EJB Object
Enterprise Bean
Appelle une
méthode du Bean
Le conteneur gère la complexité
et la cohérence d'emploi
des services techniques
Transaction
Persistance
Sécurité
Cycle de Vie
Plate-forme d'exécution J2EE
(l'ensemble des services techniques offerts)

JEE: EJB

- ***Structuration:***
  - ***Container: deux interfaces***
    - ***Home Interface*** : assure le dialogue entre EJ bean et son environnement, gère la référence du bean et son cycle de vie
    - ***Remote Interface*** : décrit les méthodes invocables par le client
  - ***Enterprise bean*** : le composant serveur qui répond aux requêtes des clients implémente les interfaces ***Home*** et ***Remote***
  - ***Application cliente:*** Utilise les interfaces Home et Remote pour utiliser l'EJB

NOTA: les interfaces peuvent être distantes (cf schéma) ou bien locales si le client est situé au sein du même container

JEE/EJB: packaging



- l'ensemble des classes du bean (.class),
- les classes d'interface ***Home*** et ***Remote*** (ou local)
- Le descripteur de déploiement **META_INF** est un ensemble de fichiers XML (***bean.xml***) et texte (fichier ***manifest.mf***)

JEE : EJB

- Développement **côté serveur**
  - 1. Ecrire une interface **Remote**
  - 2. Ecrire une interface **Home**
  - 3. Implanter les **méthodes** de l'EJB
  - 4. Ecrire un programme descripteur de **déploiement** fournissant
    - le nom de l'interface *Remote*, *Home* et de la classe d'implantation de l'EJB
    - ACL des clients et des groupes de clients autorisés
    - pour un *session bean*, dire si le bean est avec ou sans état
    - pour un *entity bean*, indiquer les variables d'instances dont la persistance est gérée par le conteneur
  - 5. Un fichier de **propriétés** décrivant les paramètres de config. du *bean*
  - 6. Un fichier ***manifest*** donnant le nom de tous les fichiers précédents

=> sert à créer un fichier ***.jar*** que l'on déploie sur les serveurs d'EJB

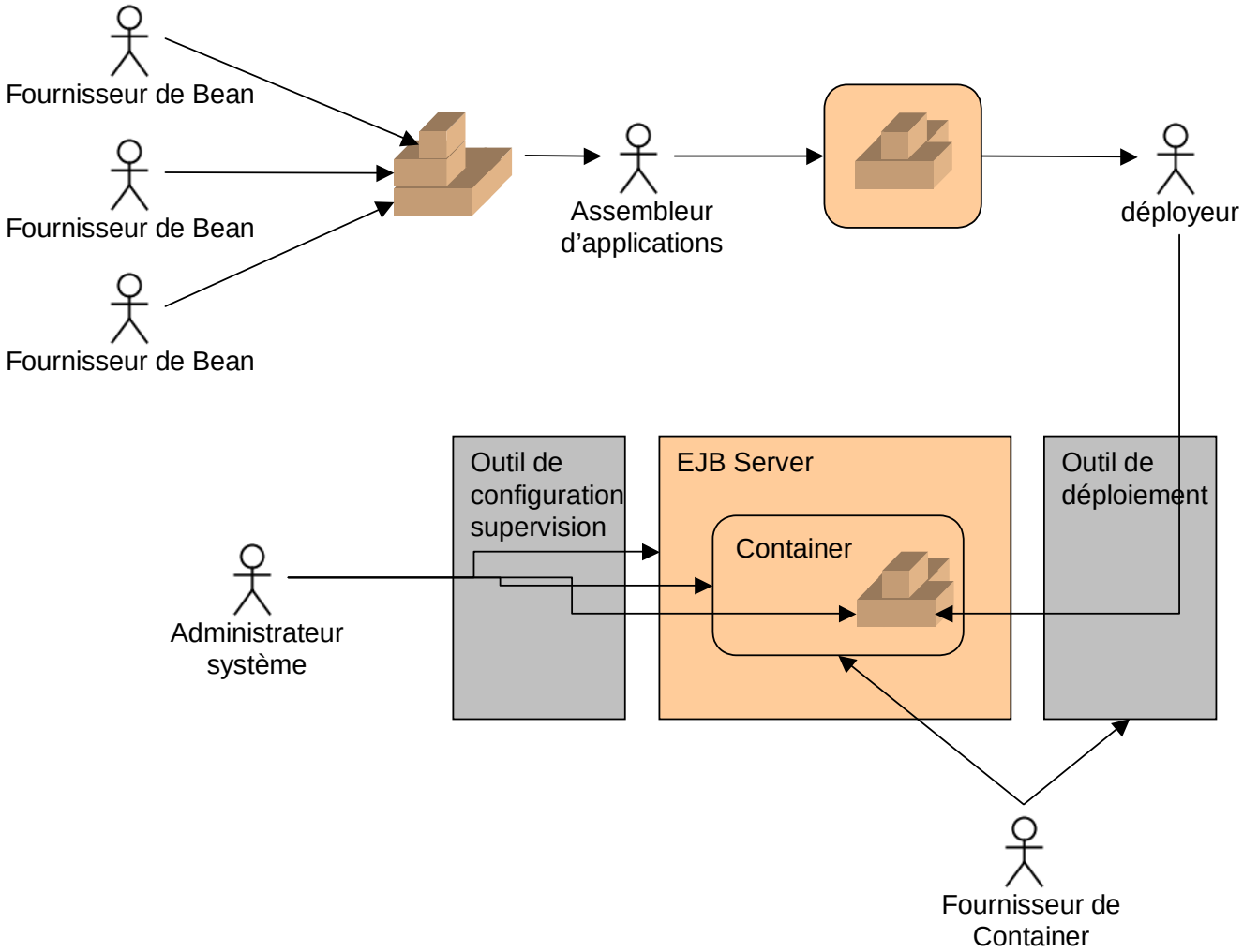
JEE : EJB

- Développement **côté client**
  - 1. Rechercher l'interface Home du *bean* par son nom via JNDI (Java Naming…) ou dans l'absolu
  - 2. Accéder au *bean:* l'interface Home permet d'accéder aux instances existantes du *bean* ou d'en créer de nouvelles => on récupère une référence sur une interface Remote
  - 3. Invocation du *bean :* on appelle les méthodes du *bean* via la référence précédente
  - 4. Fin de session
- on notifie au *bean* que la session courante est terminée

EJB: Développement

- La spécification EJB définit les rôles suivants :
  - Le fournisseur de Beans qui doit s'assurer que les Beans fournis respectent les API standards et est responsable du packaging des Beans suivant les recommandations faites par la norme (organisation du fichier ejb-jar).
  - L'assembleur d'applications
  - Le fournisseur de container EJB qui est responsable de développer le container et les services associés, ainsi que les outils permettant de déployer un Bean dans le container.
  - Le déployeur qui utilise les outils de déploiement fournis par le fournisseur de container pour déployer les Beans fournis par le fournisseur de Beans et l'assembleur d'applications.
  - Administrateur système qui est responsable de la configuration du container et du serveur, des paramètres de sécurité et de la supervision des Beans déployés.
  - Le programmeur d'application cliente, qui accède aux Beans par leurs interfaces.

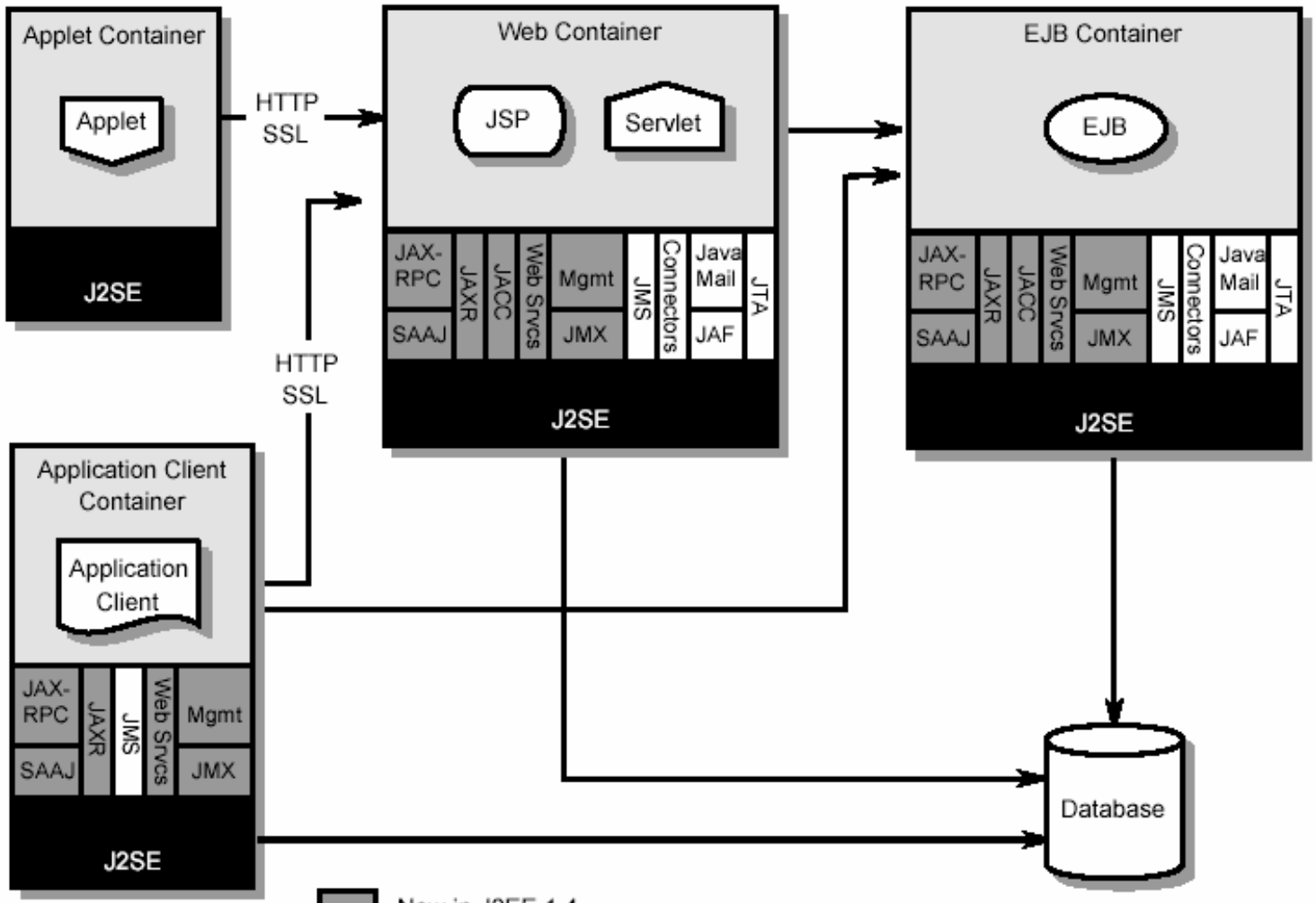
EJB: développement



JEE => API

Service	API	Description
Communication	RMI/IIOP (Remote Method Invocation)	Technologie de type ORB pour appel synchrone de service. Crée des interfaces distantes pour les communications entre applications Java. Utilise le protocole standard IIOP de CORBA.
Messages	JMS (Java Message Service)	Standard Java d'accès aux MOMs. Définit un mécanisme standard permettant à des composants d'envoyer et de recevoir des messages de façon asynchrone, pour des applications robustes et tolérantes aux pannes.
Base de données	JDBC (Java DataBase Connectivity)	Standard d'accès aux bases de données relationnelles telles que Oracle, SQL Server, Sybase, ...
Transactions	JTA/JTS (Java Transaction API/Java Transaction Service)	Tandis que J2EE fournit le support automatique des transactions, l'API JTA permet aux composants J2EE et aux clients de gérer leurs propres transactions.
Annuaire	JNDI (Java Naming and Directory Interface)	Fournit un accès aux services de nommage et d'annuaire comme DNS, NDS, LDAP et COS Naming.
Accès aux données XML	JAXP (Java API for XML Processing)	Standard Java d'implémentation des standards XML DOM, SAX, XPATH, XSLT.
Mapping objets Java-XML	JAXB (Java API for XML data Binding)	Standard Java de conversion d'objets Java vers XML et inversement.
Web Services	JAX-RPC	Standard Java de développement, déploiement et exécution de Web Services.
Authentification et Autorisation	JAAS (Java Authentication and Authorisation Service)	Standard Java pour l'authentification d'utilisateurs et la gestion des droits d'accès aux services et données.
Mail	JavaMail	Fournit la possibilité d'envoyer des e-mails.

JEE: Synthèse



JEE: Synthèse

- Marché des serveurs d'EJB
  - SUN J2EE
  - IBM WebSphere
  - BEA WebLogic Application Server
  - Oracle Application Server 4.07
  - Inprise Application Server
  - Jonas, Jboss, Jakarta, etc.

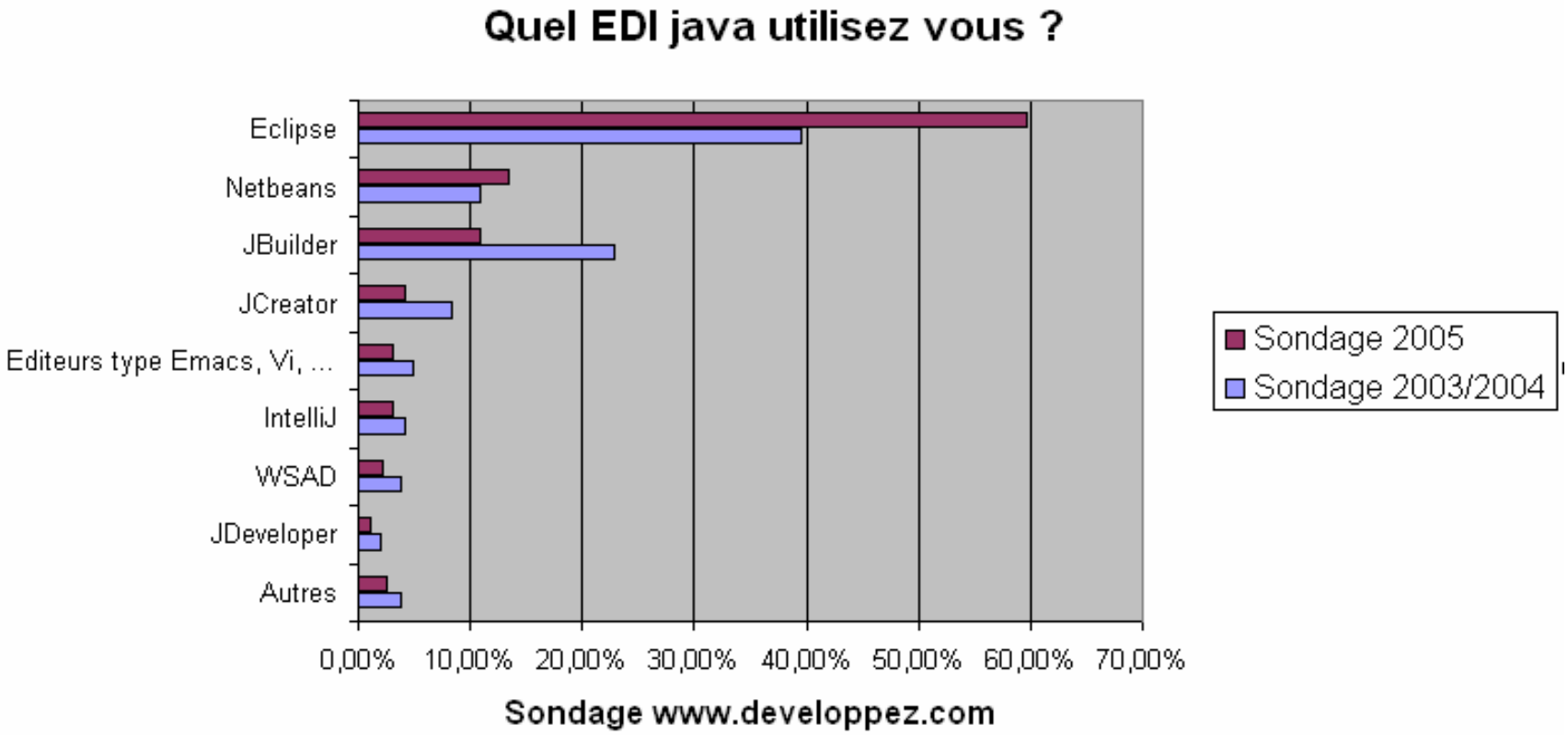
WebSphere. software

JBoss The Professional Open Source Company

The Apache Jakarta Project
http://jakarta.apache.org/

JOnAS

Outils de développement



JEE: Synthèse

- **généricité** : large spectre d'applications, (applications serveurs supportant la montée en charge grâce aux EJB et aux serveurs d'applications ou des applications clientes)
- **cible de performances** : supporte très bien la montée en charge (scalabilité) à l'exclusion des applications temps-réel.
- **robustesse / disponibilité** : utilisation en clusters, redondance
- **complexité** : oui et non... simplicité du langage mais complexité de mise en œuvre et d'architecture
- **diffusion** : Java et J2EE sont très largement répandus et disposent d'une communauté de développeurs très large,
- **standards** : l'effort de standardisation des API de J2EE est constant, au travers du consortium JCP et les produits adhérents à ces standards sont nombreux,
- **maturité** : J2EE est une technologie mature et éprouvée depuis plusieurs années, largement déployée dans le monde de l'entreprise.

- **Limites** => complexité, support de la persistance, perf pour les petites applications, IHM pour le poste client, difficultés pour l'optimisation et la supervision

Sites WEB

- Spécification
  - www.java.sun.com
- Outils de développements
  - Eclipse: www.eclipse.org
  - NETBeans: www.netbeans.org
- Serveur d'applications libres
  - JONAS: www.objectweb.org
  - JEE: www.java.sun.com
  - JBOSS: www.jboss.org
- Tutoriels
  - www.java.sun.com
  - www.developpez.com