

De l'esprit à la machine  
L'approche Professo-Académique

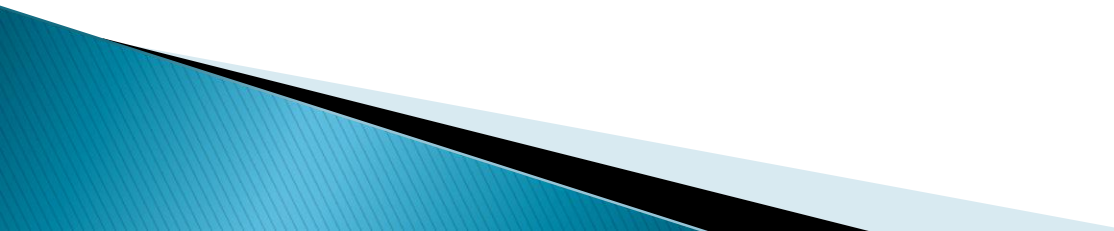
**J2EE**

**Abdelahad SATOUR**

# Séances 3–4

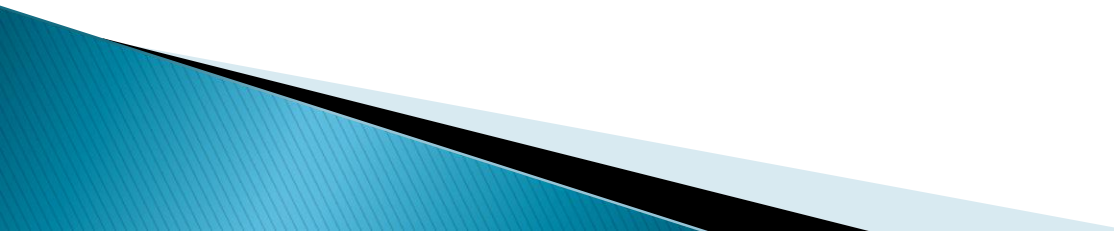
Introduction au J2EE

# Plan

- ▶ Avant-Introduction
  - ▶ Java avancé pour J2EE
  - ▶ Introduction au J2EE
  - ▶ Architecture JDK J2EE
  - ▶ Outils J2EE
  - ▶ Implémentation de J2EE
- 

# 1. Avant-Introduction

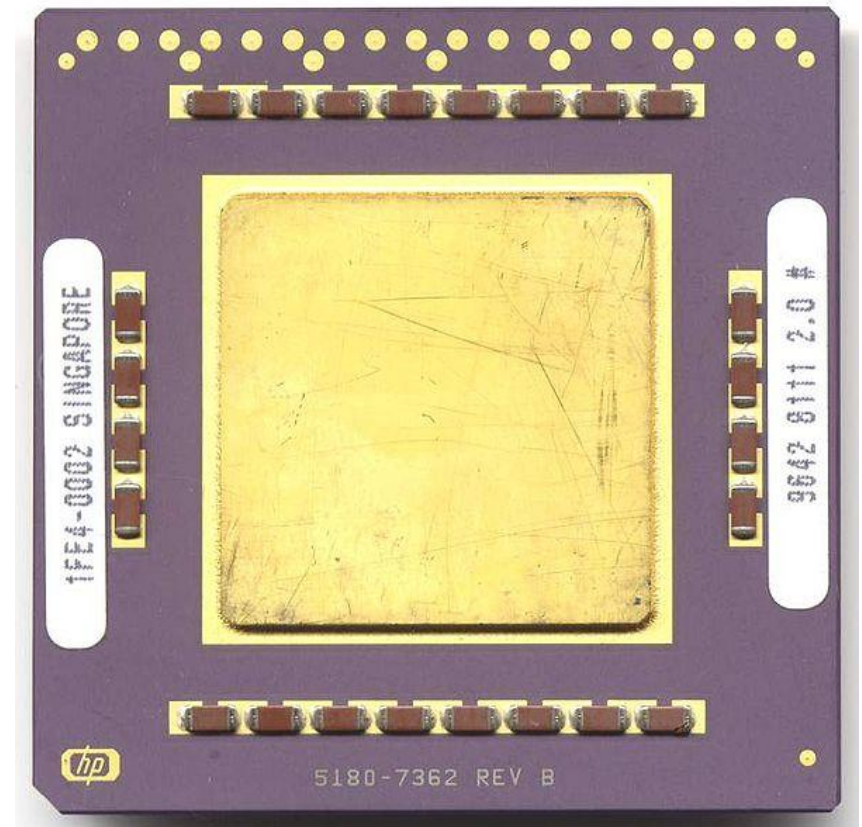
## Historique

- Java, un langage de 3ème génération
  - Issu des besoins en électronique
    - Compile once, run anywhere
    - Convergence des micro-noyaux
  - Le langage des machines à laver
  - Penser et écrire objet
  - Processeur objet
- 

# 1. Avant-Introduction

## Machine Virtuelle

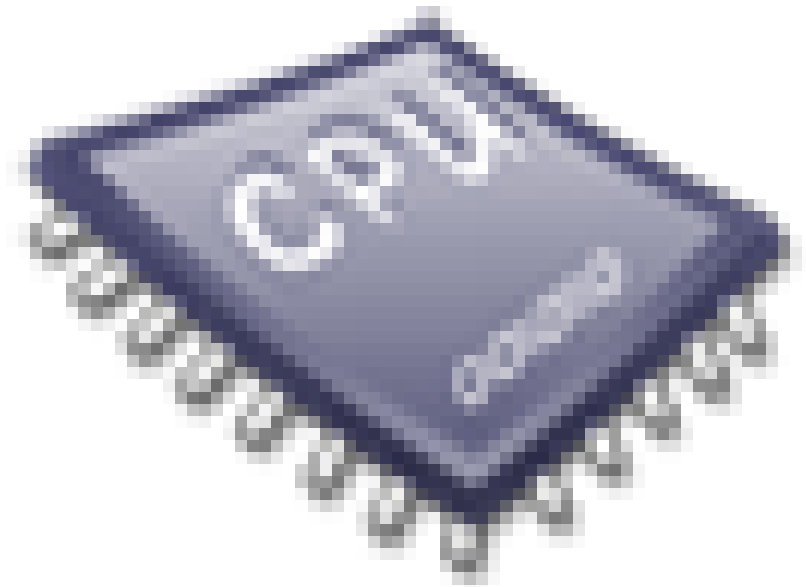
- Langage machine
- Langage C traduit en langage machine
- Trop d'architectures dissemblables
- Un langage et des librairies par projet
- Besoin de convergence



# 1. Avant-Introduction

## Machine Virtuelle

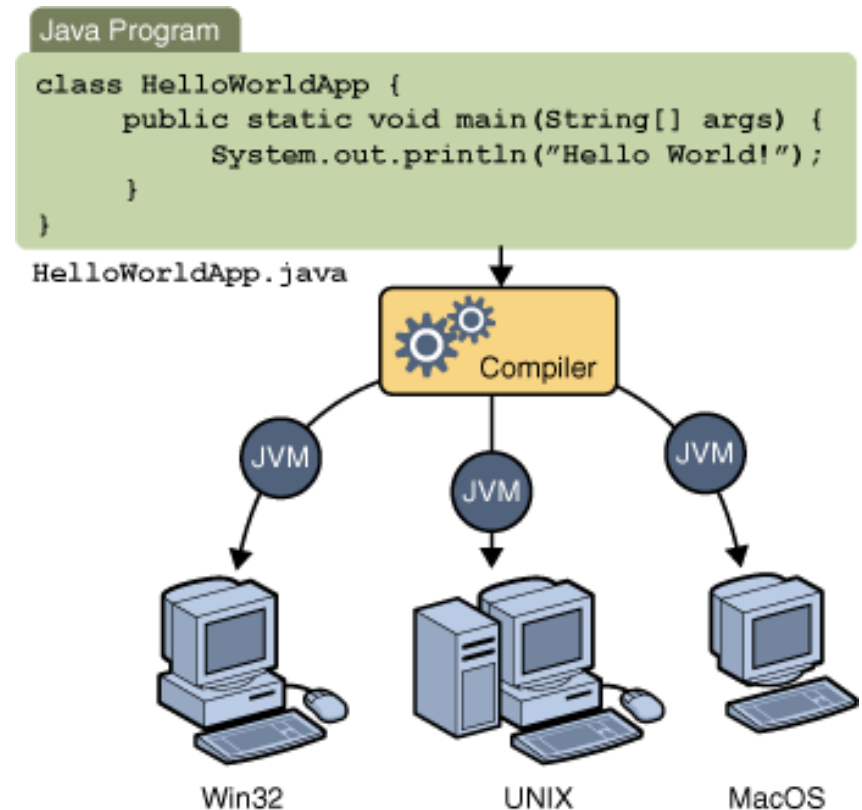
- Harmonisation de tous les systèmes
- Besoin d'une architecture commune
- Pourquoi pas l'objet



# 1. Avant-Introduction

## Machine Virtuelle

- Programmation objet
- Compilation en bytecode objet, en RPN
- Execution sur la machine virtuelle
- Une machine virtuelle pour de multiples programmes





# 1. Avant-Introduction

## Plateformes et marchés

- Un langage objet
- Une machine virtuelle sur tout device
- Un langage fortement typé
- Sécurité
  - Sécurité 1 : le langage stoppe sur erreur
  - Sécurité 2 : la machine virtuelle est sandboxée
- Développeurs et décideurs craquent
- Toutes applications sauf Graphisme real-time
  - Exemple du H263 en 1999



# 1. Avant-Introduction

## Plateformes et marchés

- Les fortune 500 ne font pas de jeu
- Le marché
  - Fiabilité
  - Prouvabilité
  - Distribution
  - Réseau
  - Maintenabilité
- C/C++ et les salaires, Python et la maintenance, Eiffel et les développeurs, Lisp, Cobol et les DBObjet



# 1. Avant-Introduction

## Plateformes et marchés

- J2SE : Java 2 Standard Edition  
2? 1.2, 1.4
- J2EE : Java 2 Enterprise Edition
- J2ME : Java 2 Micro Edition

# 1. Avant-Introduction

## Tour de J2SE

- Le langage est simple et verbeux
- Mise sur les librairies et les outils
- Une API pour chaque besoin
- Une pointe d'honneur sur les performances .. algorithmiques

# 1. Avant-Introduction

## Tour de J2SE

- Classloader
- Types de base
- Collections
- Input/Output
- Networking
- JDBC
- JAXP

Java Language									
java	javac	javadoc	apt	jar	javap	JPDA	JConsole	Java VisualVM	
Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI	
Deployment			Java Web Start				Java Plug-in		
AWT				Swing			Java 2D		
Accessibility		Drag n Drop		Input Methods		Image I/O	Print Service		Sound
IDL	JDBC		JNDI		RMI		RMI-IIOP		
Beans		Intl Support		Input/Output		JMX	JNI		Math
Networking		Override Mechanism		Security		Serialization		Extension Mechanism	XML JAXP
lang and util		Collections	Concurrency Utilities		JAR		Logging	Management	
Preferences API		Ref Objects	Reflection		Regular Expressions		Versioning	Zip	Instrumentation
Java Hotspot Client VM					Java Hotspot Server VM				

Java SE API

# 1. Avant-Introduction

## Tour de J2SE

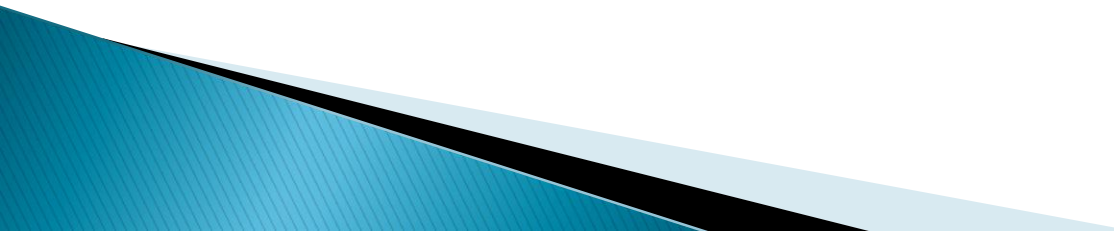
- Awt / Swing
- Concurrent
- JMX
- JNI
- Serialization
- Security
- Logging
- Regex

Java Language									
java	javac	javadoc	apt	jar	javap	JPDA	JConsole	Java VisualVM	
Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI	
Deployment			Java Web Start				Java Plug-in		
AWT				Swing			Java 2D		
Accessibility		Drag n Drop		Input Methods		Image I/O	Print Service		Sound
IDL	JDBC		JNDI		RMI		RMI-IIOP		
Beans		Intl Support		Input/Output		JMX	JNI		Math
Networking		Override Mechanism		Security		Serialization		Extension Mechanism	
XML JAXP									
lang and util		Collections	Concurrency Utilities		JAR		Logging	Management	
Preferences API		Ref Objects	Reflection		Regular Expressions		Versioning	Zip	Instrumentation
Java Hotspot Client VM					Java Hotspot Server VM				

Java SE API

# 2. Java avancé

## Environnement ... Eclipse

- JavaHome
  - Create Project
  - Change JDK eclipse Runner
  - Class
  - Rename
  - Indent
  - Comment
  - Todo
  - Move to package
- 

# 2. Java avancé

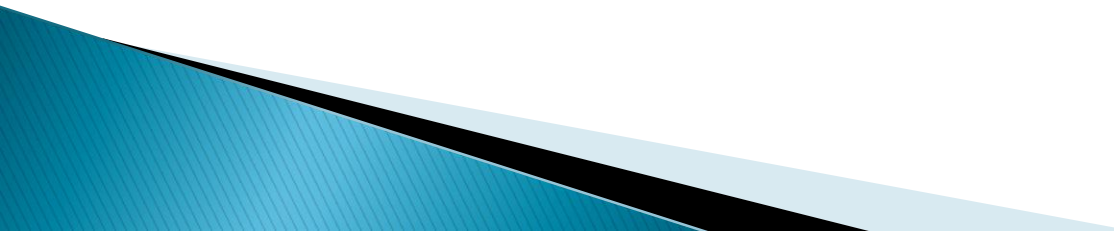
Environnement ... Eclipse

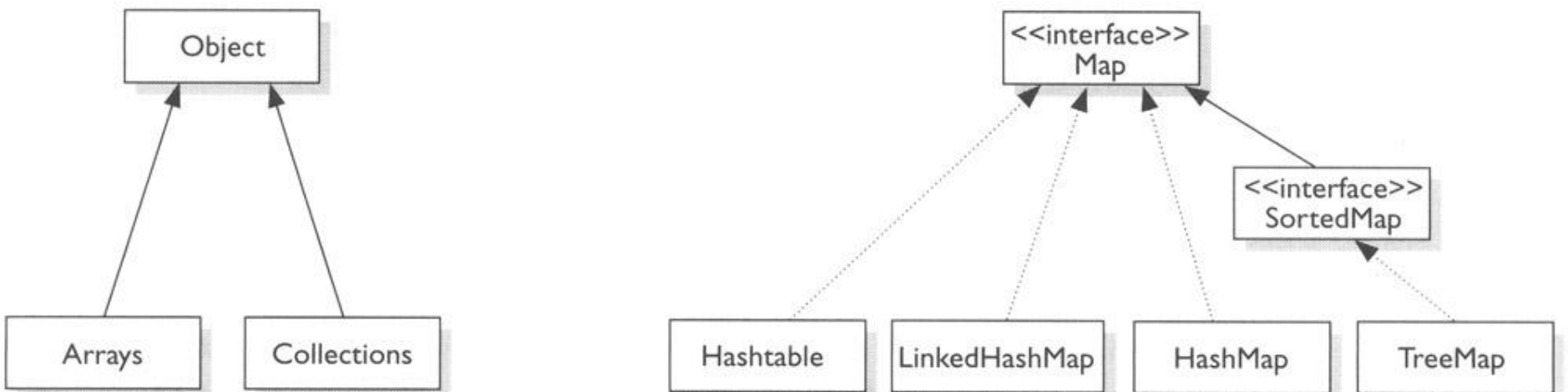
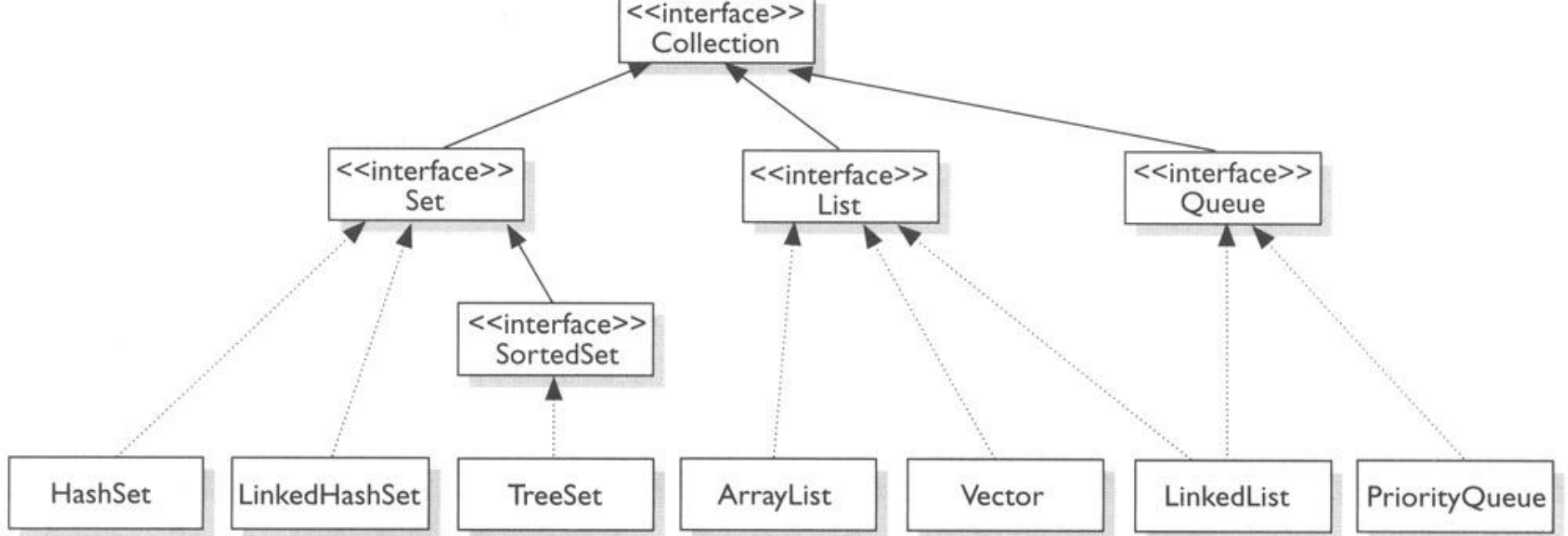
- 2 public class
- import



# 2. Java avancé

## POO

- Héritage : J2
  - Interface : J3
  - Abstract: J4
  - Final : J5, J6
- 



.....>  
implements

————>  
extends

# 2. Java avancé

## Collections

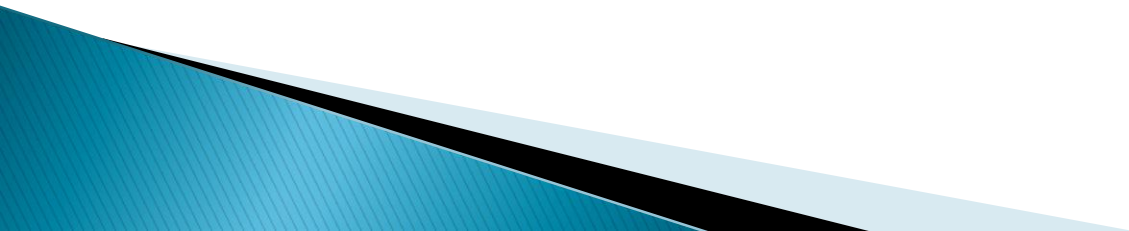
- String []
- Class[] :: Collection
- TP Collection des Collection d'une Class
- Map et HashMap
- List

Exercice

# 3. Introduction au J2EE

## Objectif

- Pourquoi une plateforme
- Pourquoi J2EE?



# 3. Introduction au J2EE

## Objectif

- ▶ Faciliter le développement de nouvelles applications à base de composants
- ▶ Intégration avec les systèmes d'information existants
- ▶ Support pour les applications « critiques » de l'entreprise
  - Disponibilité, tolérance aux pannes, montée en charge, sécurité ...

# 3. Introduction au J2EE

J2EE?

- ▶ <http://java.sun.com/javaee>  
anciennement, J2EE (Java2 Enterprise Edition)

# 3. Introduction au J2EE

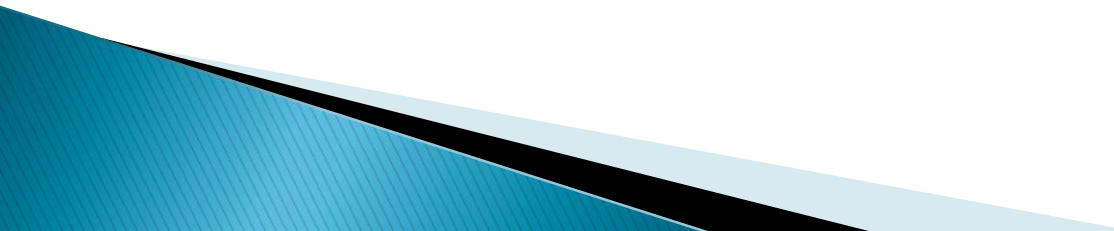
## J2EE?

- ▶ **Spécifications**  
**Les spécifications du serveur d'application**, c'est-à-dire de l'environnement d'exécution : J2EE définit finement les rôles et les interfaces pour les applications ainsi que l'environnement dans lequel elles seront exécutées. Ces recommandations permettent ainsi à des entreprises tierces de développer des serveurs d'application conformes aux spécifications ainsi définies, sans avoir à redévelopper les principaux services.
- ▶ **Des services, au travers d'API**, c'est-à-dire des extensions Java indépendantes permettant d'offrir en standard un certain nombre de fonctionnalités. Sun fournit une implémentation minimale de ces API appelée J2EE SDK (J2EE Software Development Kit).



# 3. Introduction au J2EE

## J2EE?

- ▶ Les acteurs d'une application J2EE
  - ▶ Modèle de programmation
  - ▶ Implémentation de référence
  - ▶ Suite(s) de tests
  - ▶ Label Java EE Sun (qualification de plateformes)
- 

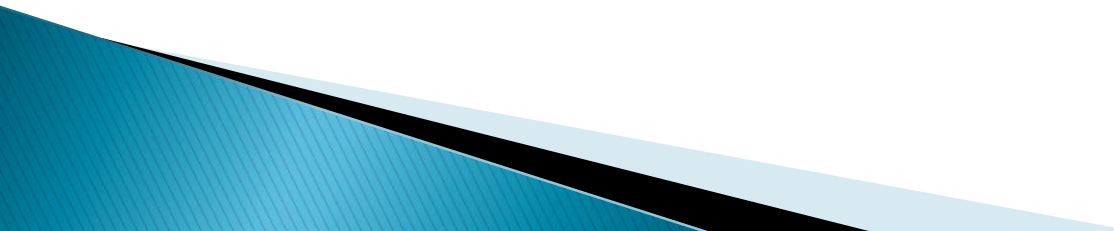
# 3. Introduction au J2EE

## J2EE?

- ▶ La réalisation d'une application basée sur l'architecture J2EE fait appel à différents types de compétences que l'on trouve rarement chez une même personne car cela va de la conception jusqu'à la supervision de l'application en passant par le développement et le déploiement. Afin de pouvoir maîtriser ce processus J2EE adopte l'approche des partages des responsabilités.
- ▶ Plus spécifiquement pour les EJB, cette approche définit plusieurs niveaux de responsabilité :
- ▶

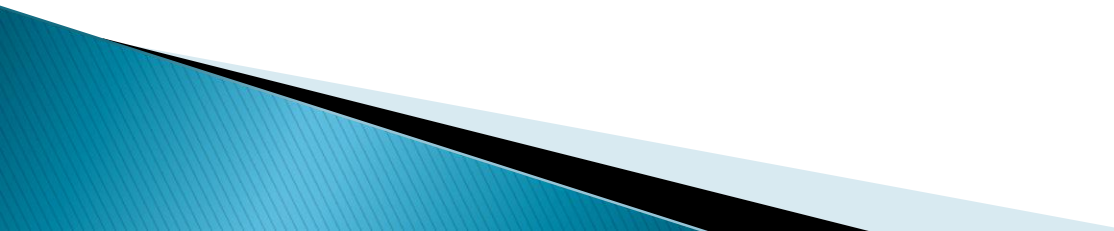
# 3. Introduction au J2EE

## J2EE?

- ▶ Le fournisseur des EJB : c'est l'acteur qui fournit des composants métiers réutilisables soit par l'achat à un fournisseur de composants, soit par développement interne ;
  - ▶ L'assembleur d'applications : l'acteur qui est capable de construire une application à partir d'EJB existants ;
  - ▶ Le déployeur : l'acteur qui récupère l'application et s'occupe de son déploiement dans un serveur d'applications ;
  - ▶ L'administrateur : l'acteur qui contrôle le fonctionnement du serveur d'application et assure la supervision des applications ;
  - ▶ Le fournisseur de conteneur : l'éditeur qui commercialise un conteneur web ou un conteneur EJB ; cet éditeur commercialise souvent un serveur d'application incluant ces conteneurs ;
  - ▶ Le fournisseur de serveur : c'est l'éditeur qui commercialise un serveur d'application (BEA, IBM etc...)
- 

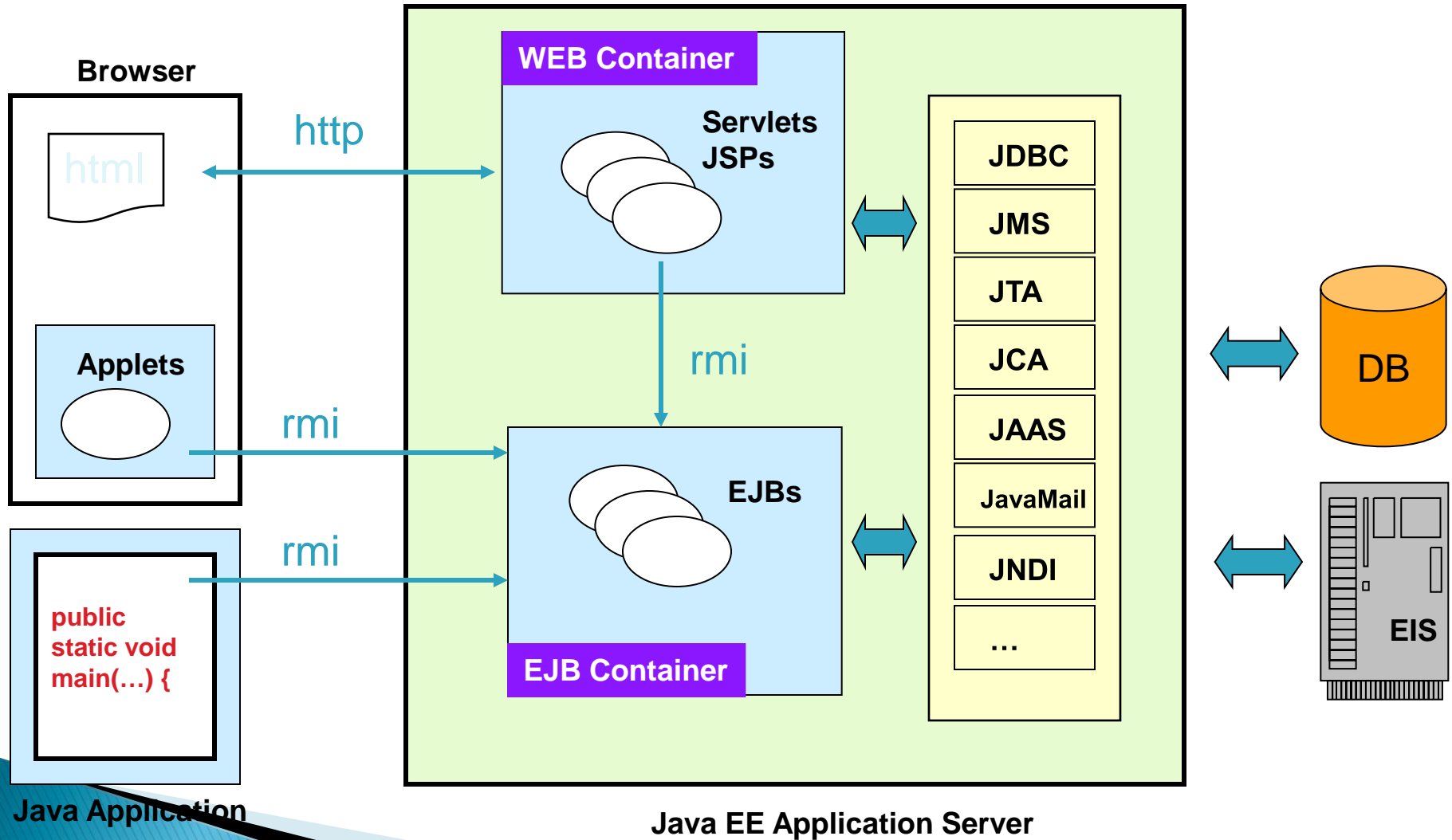
# 3. Introduction au J2EE

## J2EE?

- ▶ BEA WebLogic (haut de gamme)
  - ▶ IBM Websphere (no 1)
  - ▶ Sun Java System App Server
  - ▶ Borland Enterprise Server
  - ▶ Oracle Application Server
  - ▶ Macromedia jRun
  - ▶ SAP Web application server
  - ▶ Iona Orbix E2A
  - ▶ ...
- 

# 4. Architecture JDK

## Premier Zoom



# 4. Architecture JDK

## 1<sup>er</sup> Zoom

### ▶ Client

- Léger (Web, browser)
- Lourd (Application java, Applet...)
- Architecture orientée service (Application répartie sans présentation)

### ▶ Serveur d 'applications

- Conteneur EJB + logique métier
- Services non fonctionnels

### ▶ EIS ou Base de données



# 4. Architecture JDK

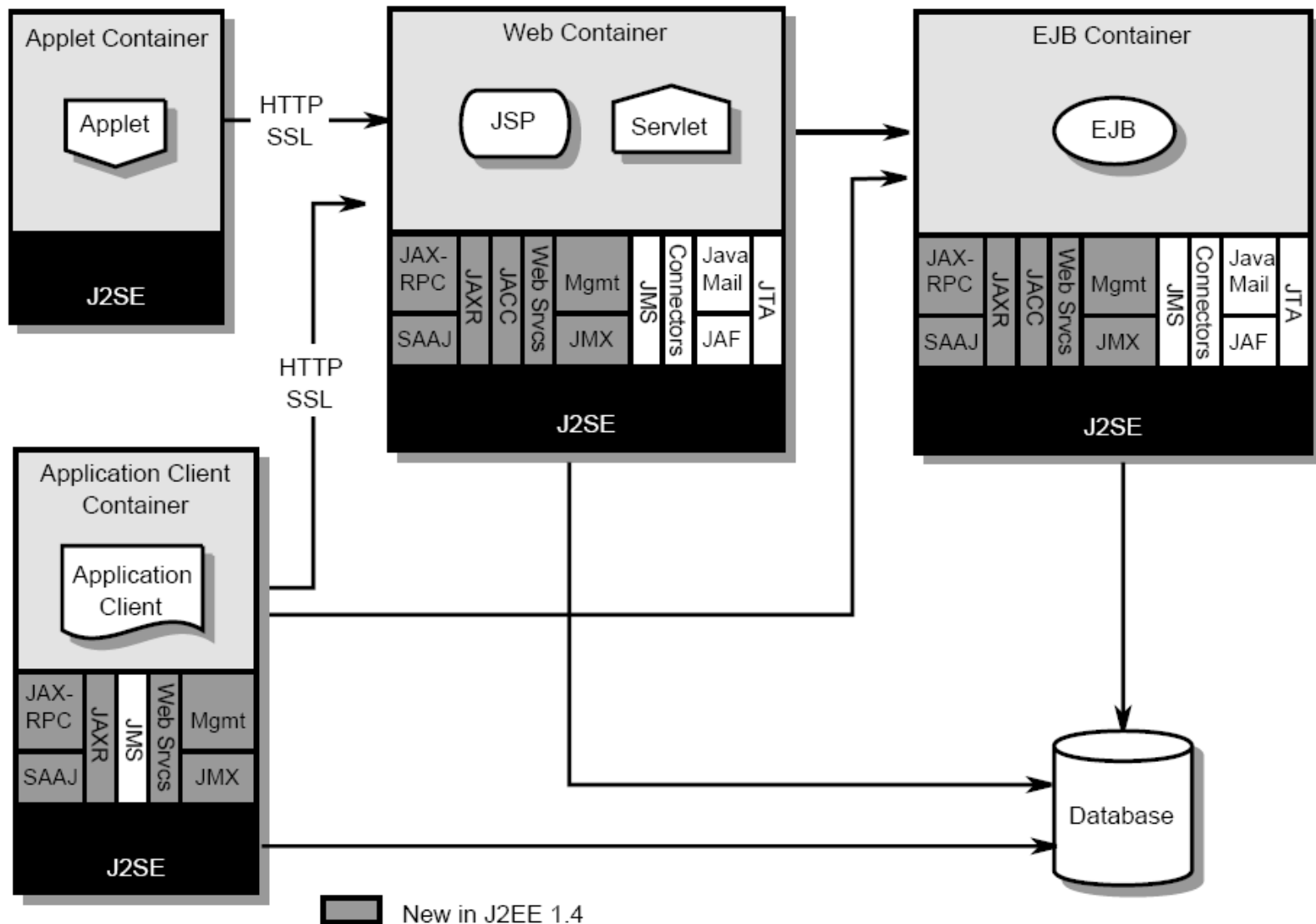
## 2<sup>ème</sup> Zoom

- ▶ J2EE décrit l'architecture d'un standard pour les serveurs d'application.
- ▶ Schéma des relations entre composants et « tiers » dans l'architecture J2EE :



# 4. Architecture JDK

## 2ème Zoom



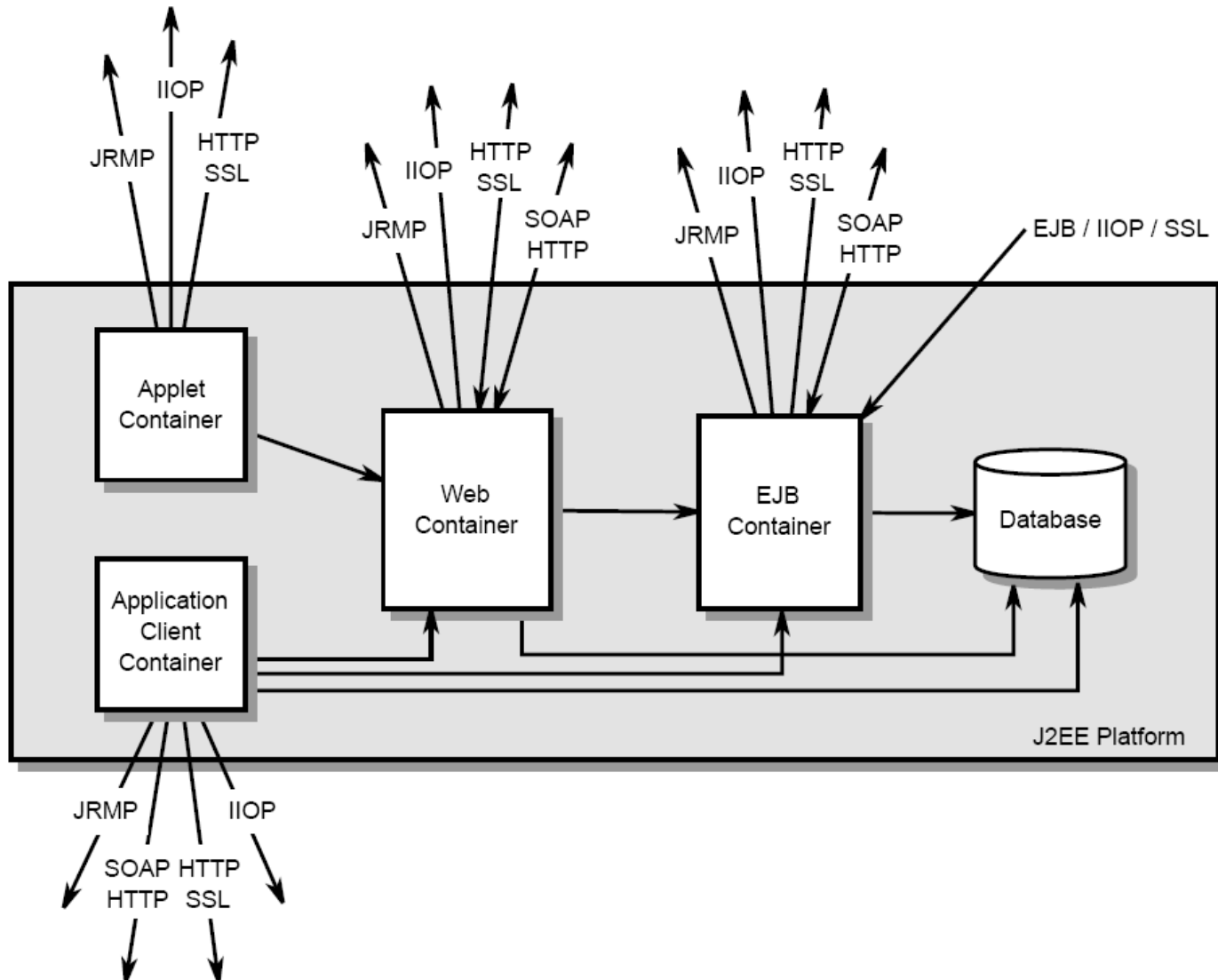
# 4. Architecture JDK

## 2<sup>ème</sup> Zoom

- ▶ Les différents rectangles définissent les conteneurs (de l'environnement d'exécution J2EE) qui fournissent les services pour les différents composants (représenter par les rectangles dans les rectangles).
- ▶ Les flèches représentent les types d'accès que le type d'application peut avoir avec les autres applications distantes. Par exemple, l'application client peut se connecter au « Web Container » par l'intermédiaire des JSP / Servlet, elle peut également se connecter à « EJB Container » ...

# 4. Architecture JDK

## 2ème Zoom



# 4. Architecture JDK

## 2<sup>ème</sup> Zoom

- ▶ RMI : Internet Inter-Orb Protocol

# 5. Outils J2EE

## Général

- ▶ Il existe 3 grands types d'outils :
  - Composants
  - Services d'infrastructures
  - Services de communications

# 5. Outils J2EE

## Général

- ▶ Il existe 3 grands types d'outils :
  - Composants
  - Services d'infrastructures
  - Services de communications

# 5. Outils J2EE

## 5.1. Composants

- ▶ On distingue, en général, 2 catégories de composants :
  - Web
  - Métiers



# 5. Outils J2EE

## 5.1.1 WEB

- ▶ Servlets

Code java exécuté sur le serveur

Equivalent du CGI

Génération de contenu Web dynamique

- ▶ JSP: Java Server Pages

Mélange de HTML/XML et de code java

Librairies d'extension (« taglibs »)

Précompilation en servlet



# 5. Outils J2EE

## 5.1.2 Métier

### ▶ **Métier – EJB (Entreprise JavaBean)**

- Il s'agit de composants spécifiques chargés des traitements des données propres à un secteur d'activité (on parle de logique métier ou de logique applicative) et de l'interfaçage avec les bases de données.
- On parle de la partie : Modèle.

# 5. Outils J2EE

## 5.2. Services d'infrastructures

- ▶ **JDBC – Java Database Connectivity**
  - C'est une API d'accès aux bases de données.
- ▶ **JNDI**
  - C'est une API d'accès aux services de nommage et aux annuaires d'entreprises tels que DNS, NIS, LDAP, etc.
- ▶ **JTA / JTS – Java Transaction Api / Java Transaction Services**
  - C'est un API définissant des interfaces standard avec un gestionnaire de transactions.
- ▶ **JCA (J2EE Connector Architecture)**
  - C'est une API de connexion au système d'information de l'entreprise, notamment aux systèmes dits «Legacy» tels que les ERP.
- ▶ **JMX (Java Management eXtension)**
  - Cette API fournit des extensions permettant de développer des applications web de supervision d'applications.

# 5. Outils J2EE

## 5.3. Services de communications

- ▶ **JAAS (Java Authentication and Authorization Service)**
  - C'est une API de gestion de l'authentification et des droits d'accès.
- ▶ **RMI (Remote Method Invocation)**
  - C'est une API permettant la communication synchrone entre objets.
- ▶ **Web services**
  - Les Web services permettent de « partager » un ensemble de méthodes qui pourront être appelées à distance. Cette technologie utilise XML, ce qui permet d'être utilisée par n'importe quel langage et n'importe quelle plateforme.
- ▶ **JMS (Java Message Service)**
  - Cette API fournit des fonctionnalités de communication asynchrone (appelées MOM pour Middleware Object Message) entre applications.
- ▶ **JavaMail**
  - C'est une API permettant l'envoi de courrier électronique.

# 6. Implémentation de J2EE

## Les serveurs d'application

- ▶ Il est avant tout indispensable de définir clairement ce qu'est un serveur d'application. En effet, une confusion règne dans les esprits quant à la notion de serveur d'application. Cette confusion a été introduite en grande partie par les éditeurs de serveurs d'application J2EE (Java2 Enterprise Edition) afin de s'approprier ce marché. La notion de serveur d'application a en effet été mélangée avec celle de serveur d'objet qui n'a absolument rien à voir

# 6. Implémentation de J2EE

## Les serveurs d'application

- ▶ Le serveur d'application est l'environnement d'exécution des applications côté serveur. Il prend en charge l'ensemble des fonctionnalités qui permettent à N clients d'utiliser une même application :

# 6. Implémentation de J2EE

## Les serveurs d'application

- ▶ 1. Gestion de la session utilisateur : N clients utilisant une même instance d'application sur le serveur, il est nécessaire que le serveur d'application puisse conserver des contextes propres à chaque utilisateur (par exemple, un panier de commandes). La plupart des serveurs d'application génèrent un identifiant unique pour chaque nouveau client et transmettent cet identifiant lors de chaque échange HTTP par URL longs, variables cachées ou cookies.

# 6. Implémentation de J2EE

## Les serveurs d'application

- ▶ 2. Gestion des montées en charge et reprise sur incident : Afin de gérer toujours plus d'utilisateurs, le serveur d'application doit pouvoir se déployer sur plusieurs machines et éventuellement offrir des possibilités de reprise sur incident (même si dans la grande majorité des cas, on se contente d'une gestion des montées en charge au niveau réseau – boîtier de répartition, DNS round-robin, reverse proxy ...).



# 6. Implémentation de J2EE

## Les serveurs d'application

- ▶ 3. Ouverture sur de multiples sources de données : C'est le serveur d'application qui rend accessible les données des applications du système d'information. Il doit donc pouvoir accéder à de nombreuses sources de données. On s'attend également à ce qu'il fournisse des mécanismes performants comme le pooling de connexion base de données.

# 6. Implémentation de J2EE

## Les serveurs d'application

- ▶ Le serveur d'application est donc indispensable si l'on souhaite éviter de re-développer l'ensemble de ces fonctionnalités (cas des GGI). Les moteurs JSP/Servlets, Microsoft ASP, Cold Fusion, PHP ... sont à ce titre des serveurs d'application (même si il sont intégrés au ServeurWeb PHP/ASP).

# 6. Implémentation de J2EE

## Serveur d'objet

- ▶ Pour aborder la notion de serveur d'objets, il faut comprendre qu'il existe deux méthodes pour accéder aux données et aux traitements :

# 6. Implémentation de J2EE

## Serveur d'objet

- ▶ 1. La première consiste à accéder directement aux sources de données. Cette méthode de programmation n'empêche en aucun cas de structurer ses développements.

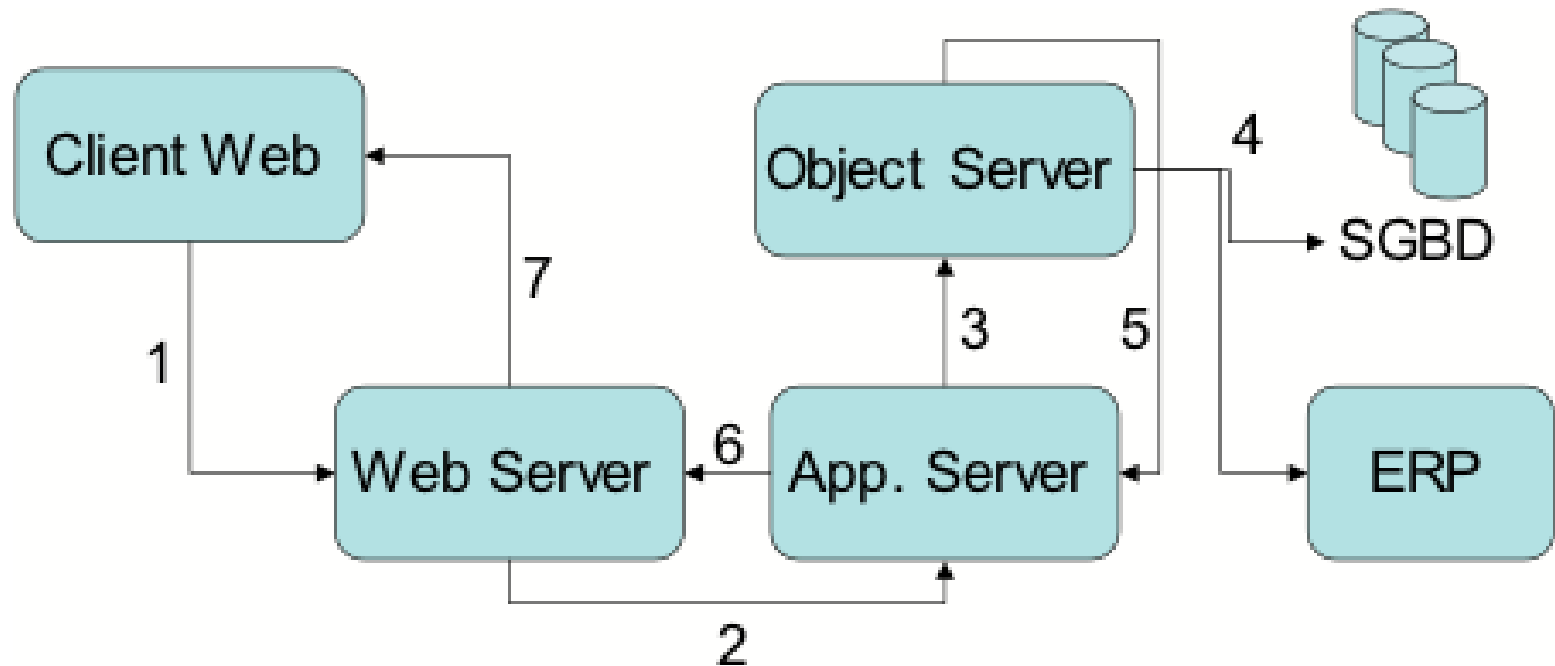
# 6. Implémentation de J2EE

## Serveur d'objet

- ▶ 2. La deuxième méthode consiste à s'appuyer sur des objets métier (client, fournisseur ...) afin de masquer la complexité d'accès aux données. Un objet AssuréSocial possédera par exemple une méthode débit() et une méthode crédit () qui à chaque appel iront modifier les données dans une ERP (Entreprise Resource Planning), un système de CRM (Customer Relation Ship Managment) ou une base de données.

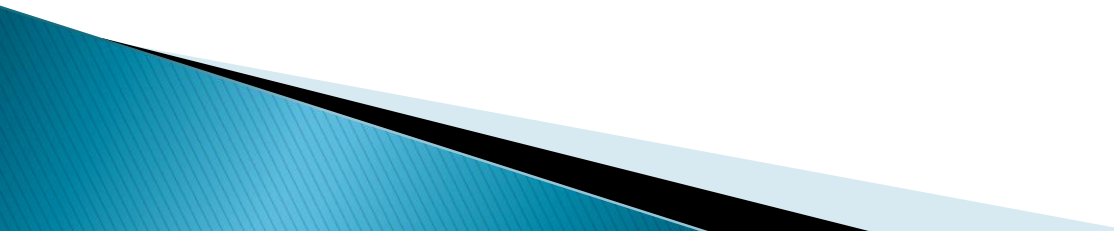
# 6. Implémentation de J2EE

## Serveur d'objet



# 6. Implémentation de J2EE

## Serveur d'objet

- ▶ Requête du client
  - ▶ Le serveur web passe la requête au serveur d'application
  - ▶ Le serveur d'application traite la requête par des appels au serveur d'objets
  - ▶ Le serveur d'objet traite les données avec les bases de données (en tout genre)
  - ▶ Le serveur d'objet retourne les objets au serveur d'application
  - ▶ Le serveur d'application renvoie le résultat au serveur web
  - ▶ Le serveur web fait suivre le résultat au client
- 

# 6. Implémentation de J2EE

## Serveur d'objet

- ▶ Pour gérer ces objets, un environnement d'exploitation est nécessaire : le serveur d'objets. Ce serveur d'objets va devoir fournir des services tout à fait différents de ceux des serveurs d'application :
  - Gestion de la persistance des objets,
  - Gestion des transactions objets métier
  - Gestion des montées en charge : ici les mécanismes n'ont rien à voir avec ceux mis en oeuvre pour un serveur d'application. Il faut pouvoir assurer la transparence de localisation, l'instanciation, ... des objets métier ...



# 6. Implémentation de J2EE

## Serveur d'objet

- ▶ Les points clés d'une architecture sont les capacités transactionnelles du serveur d'application à délivrer des pages et à intégrer une montée en charge... L'ergonomie au sens large est un autre point clé. Les choix de design doivent prendre en compte les contraintes du Web (taille des images, ...).
- ▶ Le marché offre trois familles de solutions de développement pour les serveurs d'application.
- ▶ Les solutions de scripting peuvent être simples et productives mais plutôt orientées vers les sites jetables, de type événementiel. Un site en ASP, PHP 3 ou ColdFusion peut être développé très rapidement ; par contre, sa maintenance est compliquée voire quasi-impossible.
- ▶ Les solutions orientées objets techniques permettent de factoriser le code sans rentrer dans la complexité des objets métier. Il est important d'imposer des règles de développement précises à ses équipes et prestataires. Les développements JSP/Servlets/JavaBeans, PHP4/5, ASP/DCOM (et ASP.Net/DCOM) permettent de tels développements.

# 6. Implémentation de J2EE

## Serveur d'objet

- ▶ Les solutions orientées métier sont plus complexes et plus coûteuses à mettre en oeuvre. Elles nécessitent la mise en place de serveur d'objets. On retrouve principalement sur ce marché les serveurs d'EJB libres et propriétaires.
- ▶ Pour ces trois familles de solutions, des produits Open Source existent et sont de plus en plus adoptés dans les administrations et entreprises (TomCat, JBoss, JonAS...).