J2EE

Module Java

- Vue d'ensemble du langage Java
- Le langage Java : syntaxe et sémantique
- Programmation multi-tâche : les threads
- Accéder aux bases de données
- Composants réutilisables : le modèle MVC
- Développement Client/Serveur
- Présentation de l'IDE VisualAge
- Les serveurs d'applications J2EE
- Les Enterprise JavaBeans
- Ré-ingénierie d'applications Java

Deruelle Laurent

Présentation de J2EE

- Java 2 Platform, Enterprise Edition est sorti officiellement à Java One en 1998 en même temps que les Enterprise JavaBeans.
- Il est né de besoins grandissant des entreprises pour développer des applications complexes distribuées et ouvertes sur l 'Internet.

Deruelle Laurent

Présentation de J2EE

- Java 2 Platform, Enterprise Edition inclut les API fondamentales Java suivantes:
  - Les Enterprise JavaBeans (EJB) : composants métiers
  - Les JavaServer Pages (JSP) et les Servlets : composants Web
  - Java DataBase Connectivity (JDBC) pour l 'accès aux bases de données relationnelles.

Deruelle Laurent

Présentation de J2EE

- Java Transaction Service (JTS) permet d 'accéder à un service de transactions répartis.
- L 'API Java Transaction (JTA) fournit une démarcation des transactions dans une application.
- Java Message Service (JMS) : pour accéder à divers services de messageries.

Présentation de J2EE

- Java Naming and Directory Interface (JNDI) fournit un accès aux services de dénomination, DNS, LDAP.
- Remote Method Invocation (RMI) sur Internet Inter-ORB Protocol (IIOP) permet une invocation de méthodes à distance au-dessus du protocole IIOP de CORBA.

J2EE - Présentation

- Avec J2EE, Sun essaie de faire de Java un langage à part entière, mais surtout une plate-forme viable de développement dans le cadre de l 'entreprise
- J2EE n'est pas un simple regroupement d 'APIs
- Elle définit également un environnement de programmation basé sur une architecture d'exécution
- Un des objectifs : faciliter la vie des développeurs en permettant l'encapsulation de la complexité inhérente aux environnements distribués dans une architecture basée sur les **conteneurs**
- Le programmeur n'a plus à se soucier que de la rédaction de la logique métier de son application

Deruelle Laurent

La plate-forme J2EE

- Environnement d'exécution de J2EE
  - J2EE se contente de regrouper un certain nombre d'API, mais il présente également la caractéristique remarquable de faire abstraction de l'**infrastructure d'exécution**
    - => Juste une spécification, ensuite implantée par les éditeurs logiciels qui mettent au point les serveurs d'applications
  - Informatique distribuée "traditionnelle" = souvent problèmes liés non pas à la logique propre à l'application mais à la mise en œuvre de services complexes (threading, transactions, sécurité…)
  - J2EE introduit la notion de **conteneur**, et via les **API J2EE**, il élabore un **contrat** entre le conteneur et les applications
  - C'est le vendeur du conteneur qui se charge de mettre en œuvre les services pour les développeurs d'applications J2EE, dans le respect des **standards**

Technologies J2EE

- Technologies composants
  - Contient la partie la plus importante de l'application : la logique métier
  - 3 types de composants : les JSP, les Servlets et les EJB
- Technologies de services
  - fournissent aux composants de l'application des services connexes leur permettant de fonctionner en toute efficacité
- Technologies de communication
  - Quasiment transparentes pour le programmeur d'applications, elles prévoient les mécanismes de communication entre les différentes parties de l'application, qu'elles soient locales ou distantes

Technologies J2EE

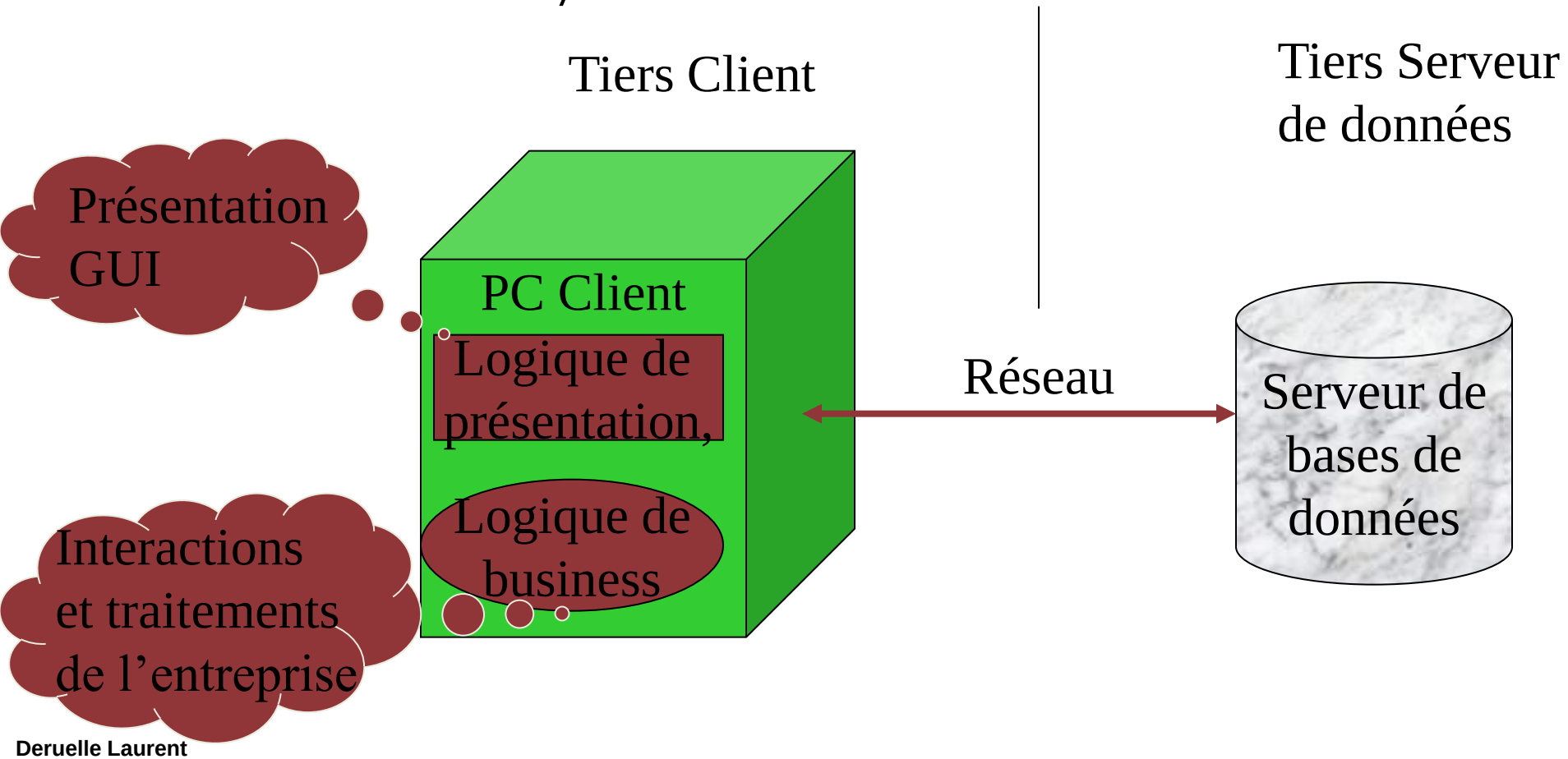
- Technologies composants
  - Quelle que soit l'application, le but essentiel est de modéliser la logique métier nécessaire en utilisant les composant, unités réutilisables au niveau applicatif
    - Donc, même si le conteneur est en mesure de fournir l'environnement d'exécution, la couche de communication ainsi que de nombreux services, la création des composants applicatifs incombe au développeur
  - Toutefois, ces composants dépendront de leurs conteneurs pour de nombreux services, tels que la gestion du cycle de vie, le threading, la sécurité...
  - => Ceci permet de se consacrer uniquement aux méthodes métier sans s'appesantir sur la sémantique de bas niveau du conteneur

Avantages de J2EE

- Standard de plate-forme Java
  - pour les éditeurs de Systèmes d 'Entreprise
  - implémenter des produits compatibles
  - bénéficier des avantages de la technologie composant
  - se concentrer sur le business au lieu de résoudre les problèmes d 'intégration.

Architecture deux-tiers

- Applications centrées sur des gros systèmes.
- 2-tiers = client/serveur

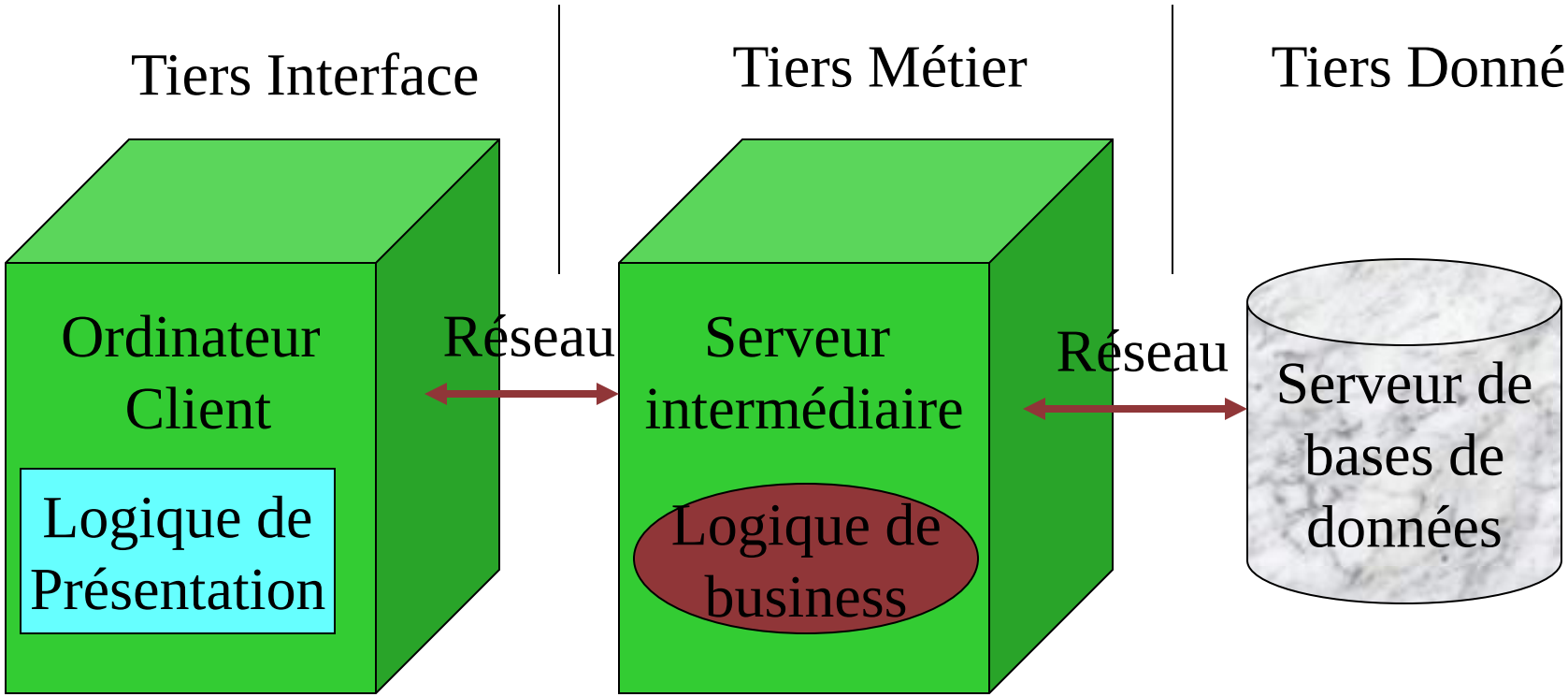


Limites de l ’architecture

- Intégrité des bases de données facilement compromise.
- Administration difficile lorsque la logique métier doit être modifiée.
- Transgression de la sécurité en altérant un processus business.
- Application liée à un seul type de présentation

Architecture à trois niveaux

- Séparation de la logique de présentation et de la logique métier.



Avantages de l ’architecture à trois .

- Améliore les montée en charge et le nombre de connections.
- Meilleure performance, sécurité et gestion de l ’application.

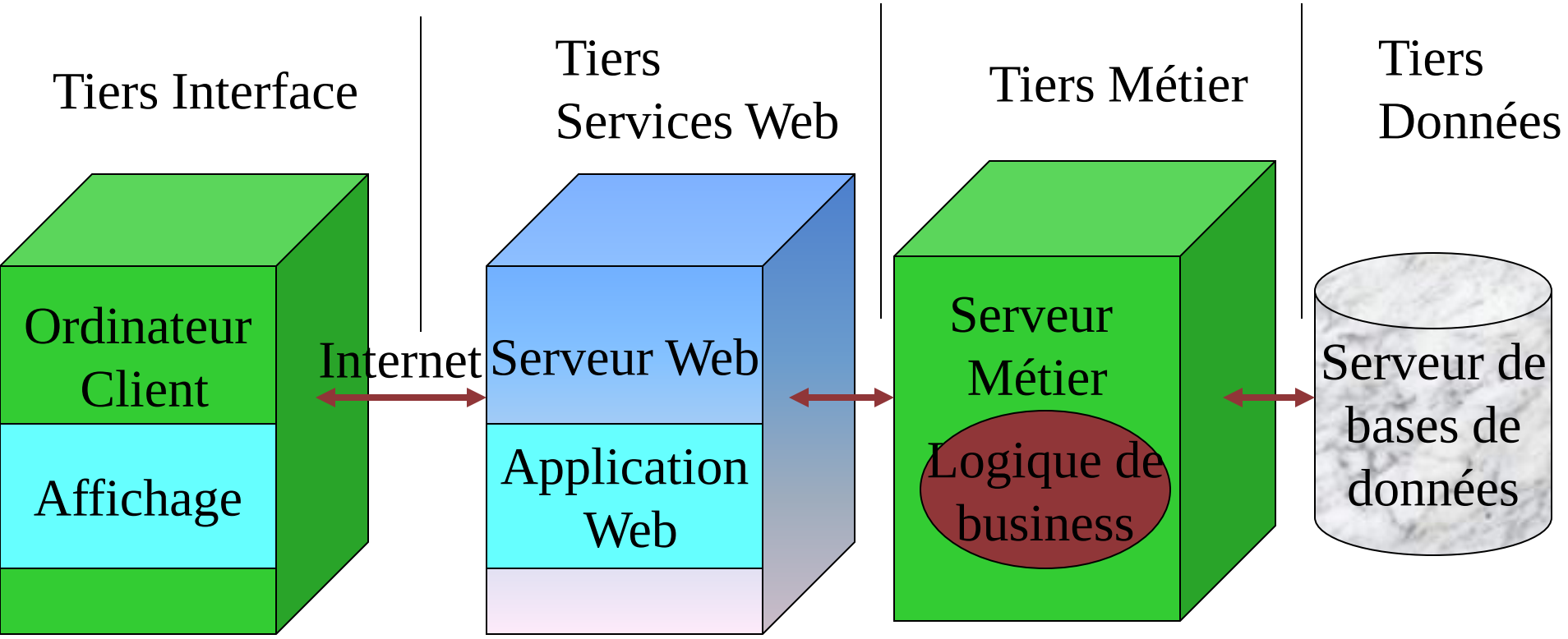
Deruelle Laurent

Limites de l 'architecture à trois

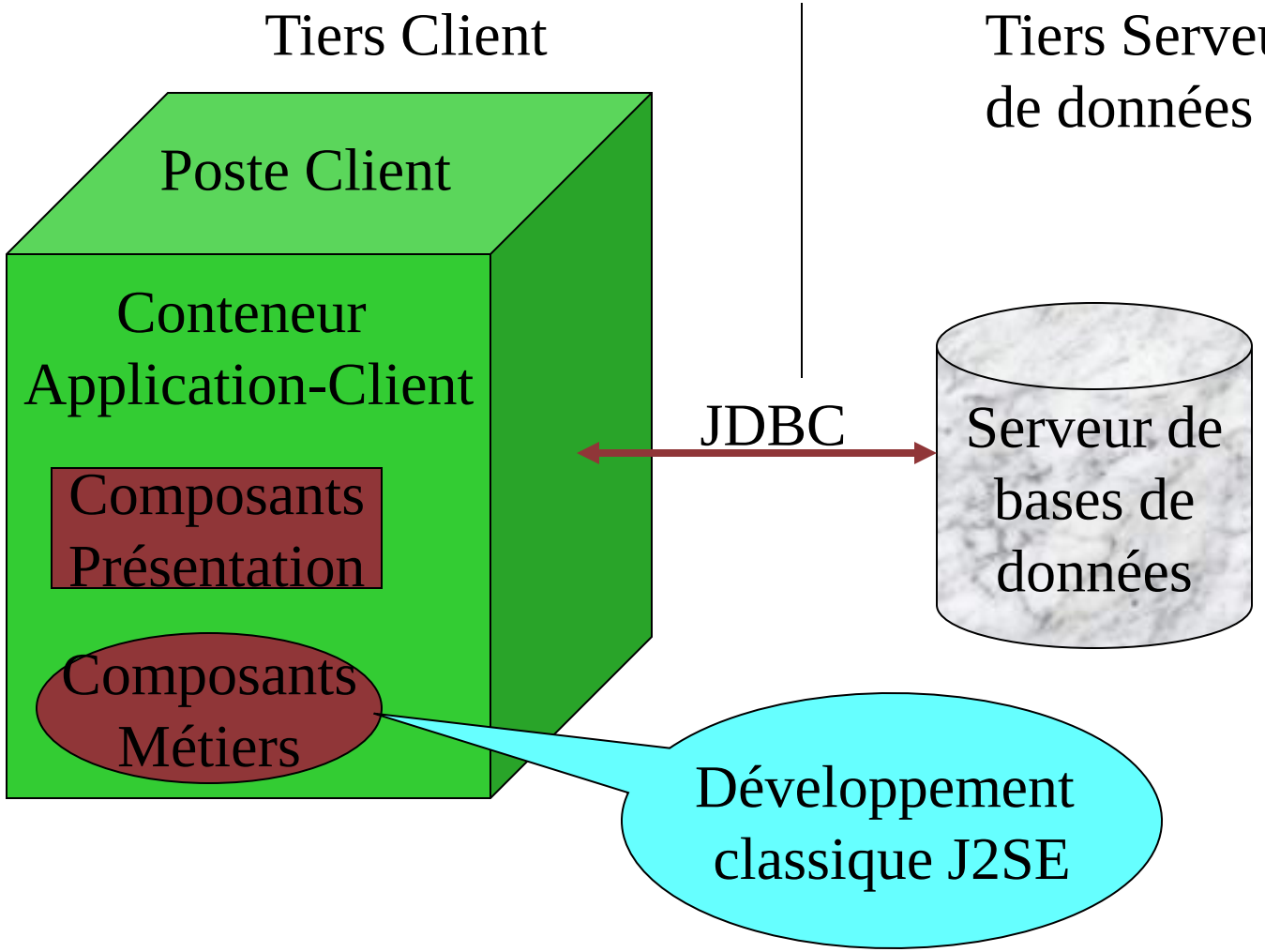
- Plus complexe de développer ce type d 'application:
  - gestion de la répartition, multithreading,
  - gestion de la sécurité.
- Difficulté d 'intégration entre les systèmes 3-tiers.
- L 'architecture *classique* 3-tiers n'est pas prévue pour le Web

Architecture à quatre niveaux

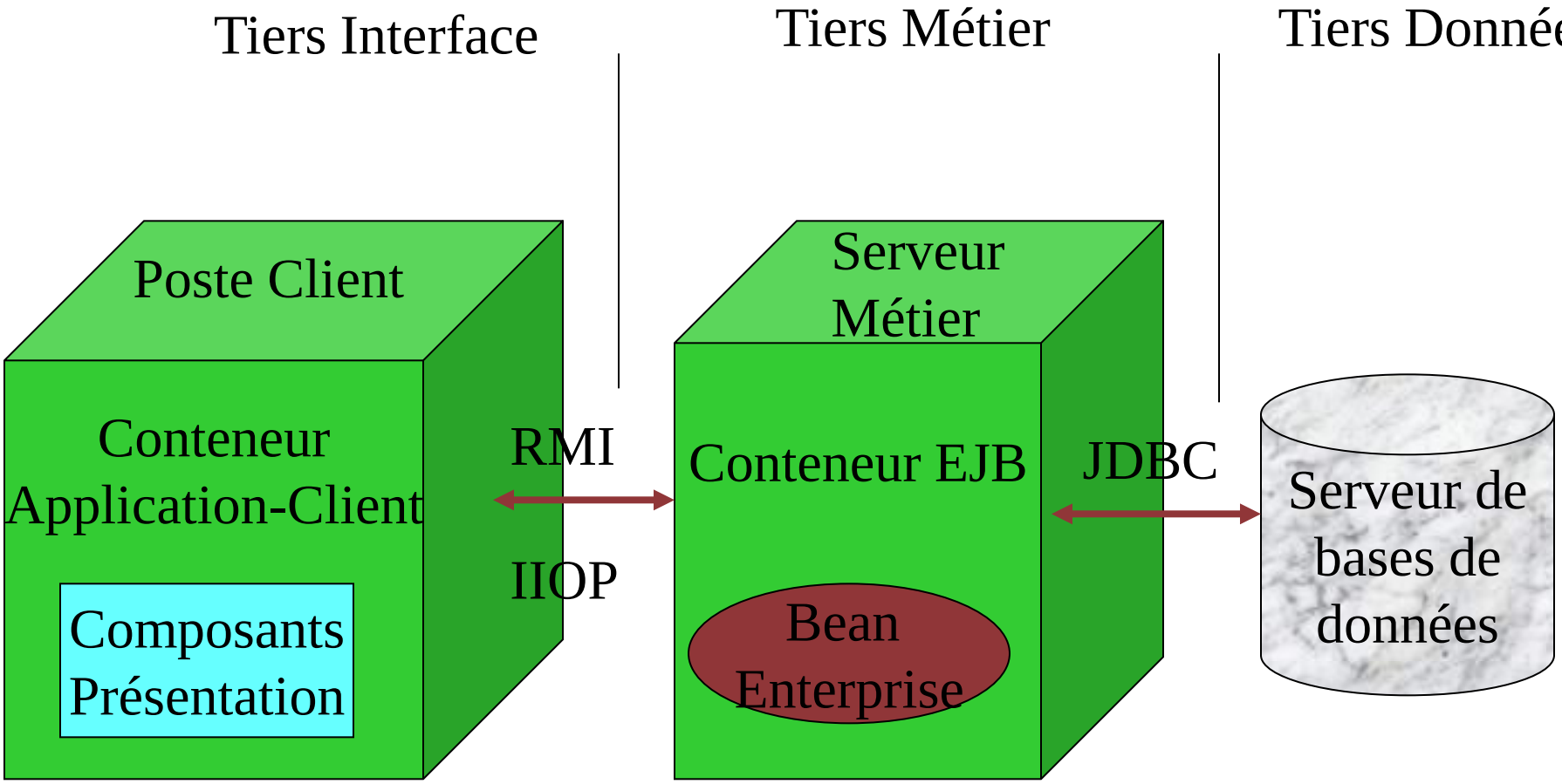
- Introduction et développement d 'applications orientées Web: J2EE



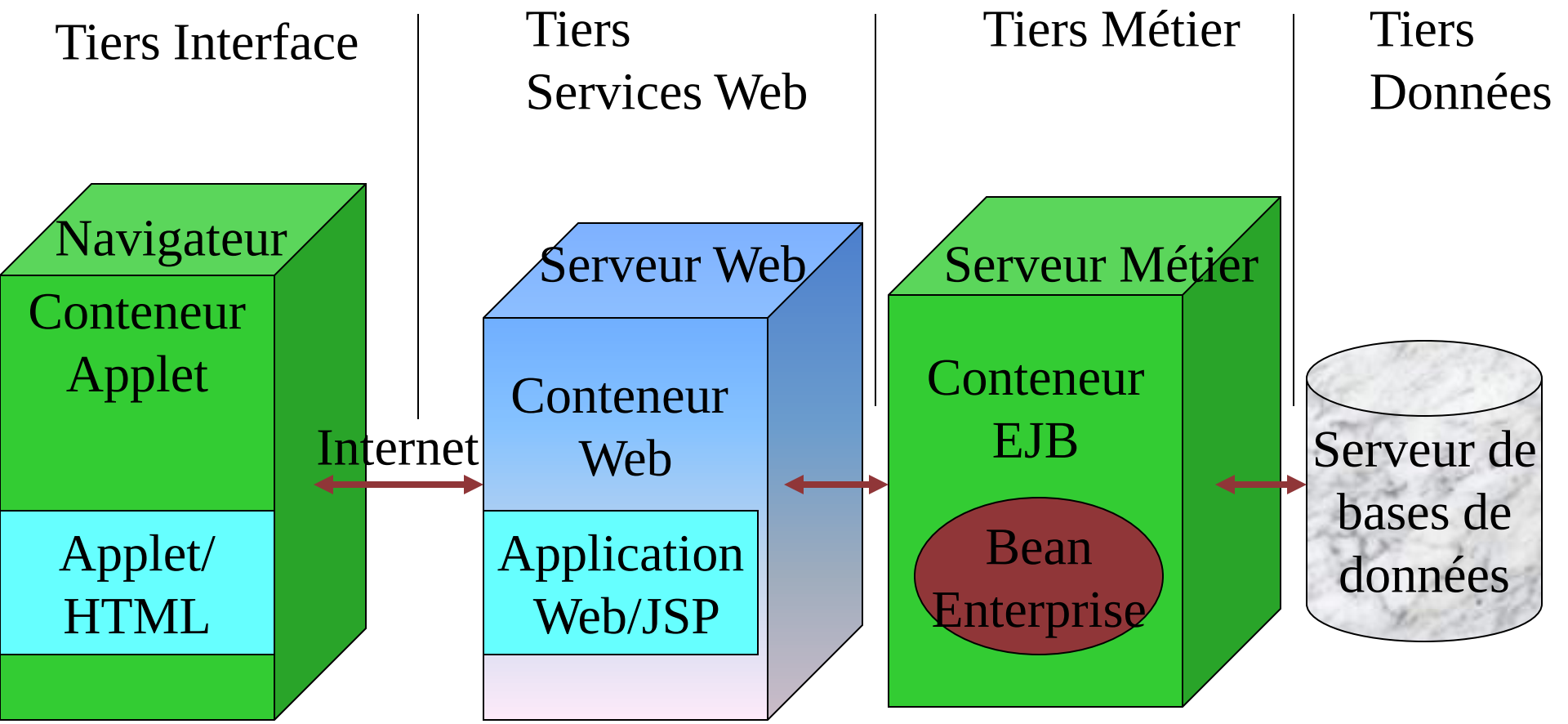
Modèle J2EE pour les architectures à deux niveaux



Modèle J2EE pour les architectures à trois niveaux



Modèle J2EE pour les architectures à trois niveaux

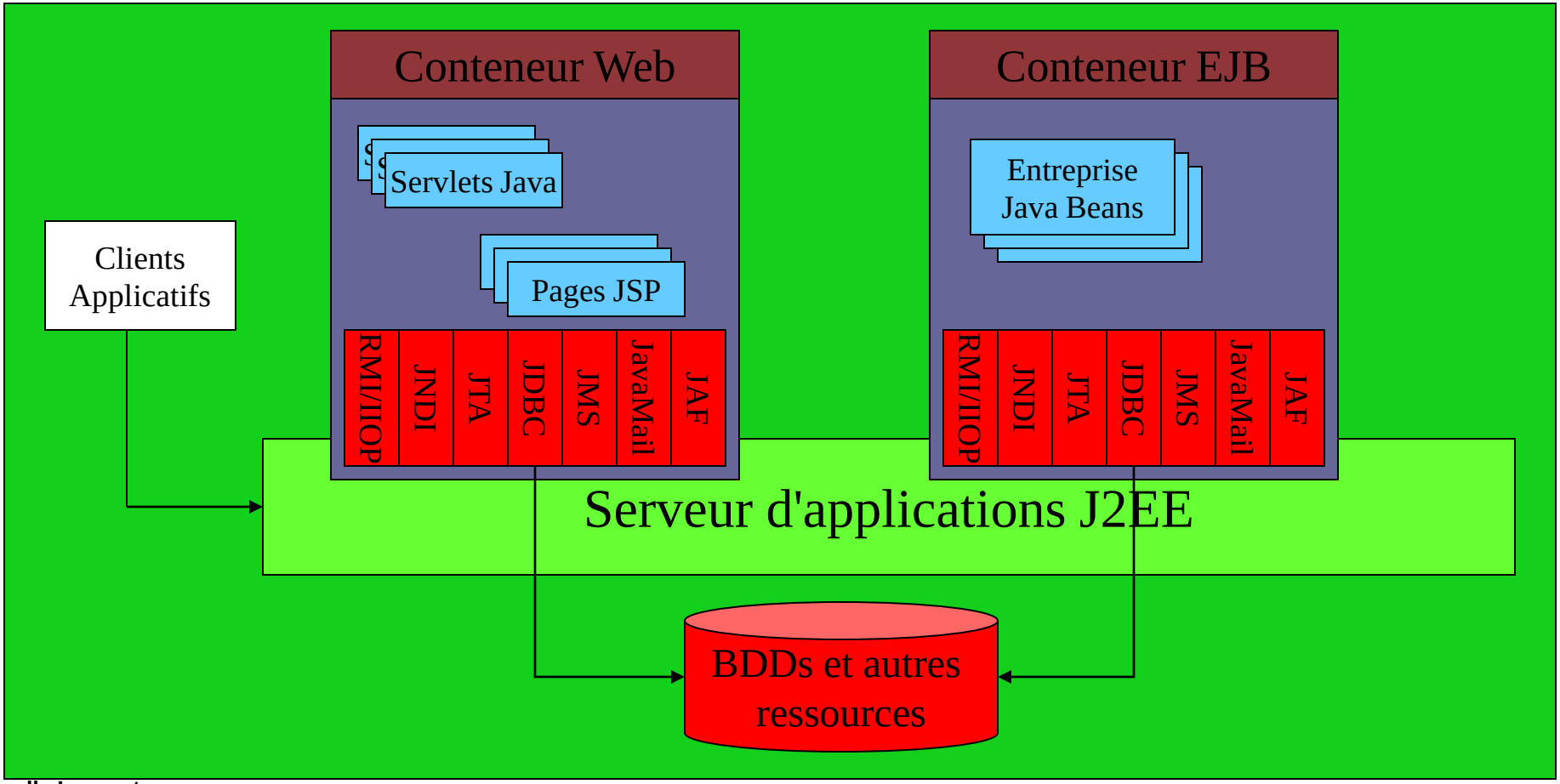


Architecture J2EE

- La plate-forme J2EE est constituée de quatre environnements de programmation baptisés *containers* :
  - le container EJB
  - le container Web
  - le container Application-Client
  - le container Applet

Architecture J2EE - Conteneurs

- Un conteneur J2EE est un environnement d'exécution chargé de gérer des composants applicatifs et de donner accès aux API J2EE



Le container Application-Client

- Il fournit l 'environnement d 'exécution des clients J2EE, comportant des interfaces basées sur JFC/Swing.
- Il s 'agit essentiellement de Java 2 Standard Edition (J2SE).

Deruelle Laurent

Le container Applet

- Il permet l 'exécution des *Applets* Java.
- Cet environnement est généralement intégré au sein du navigateur Web.

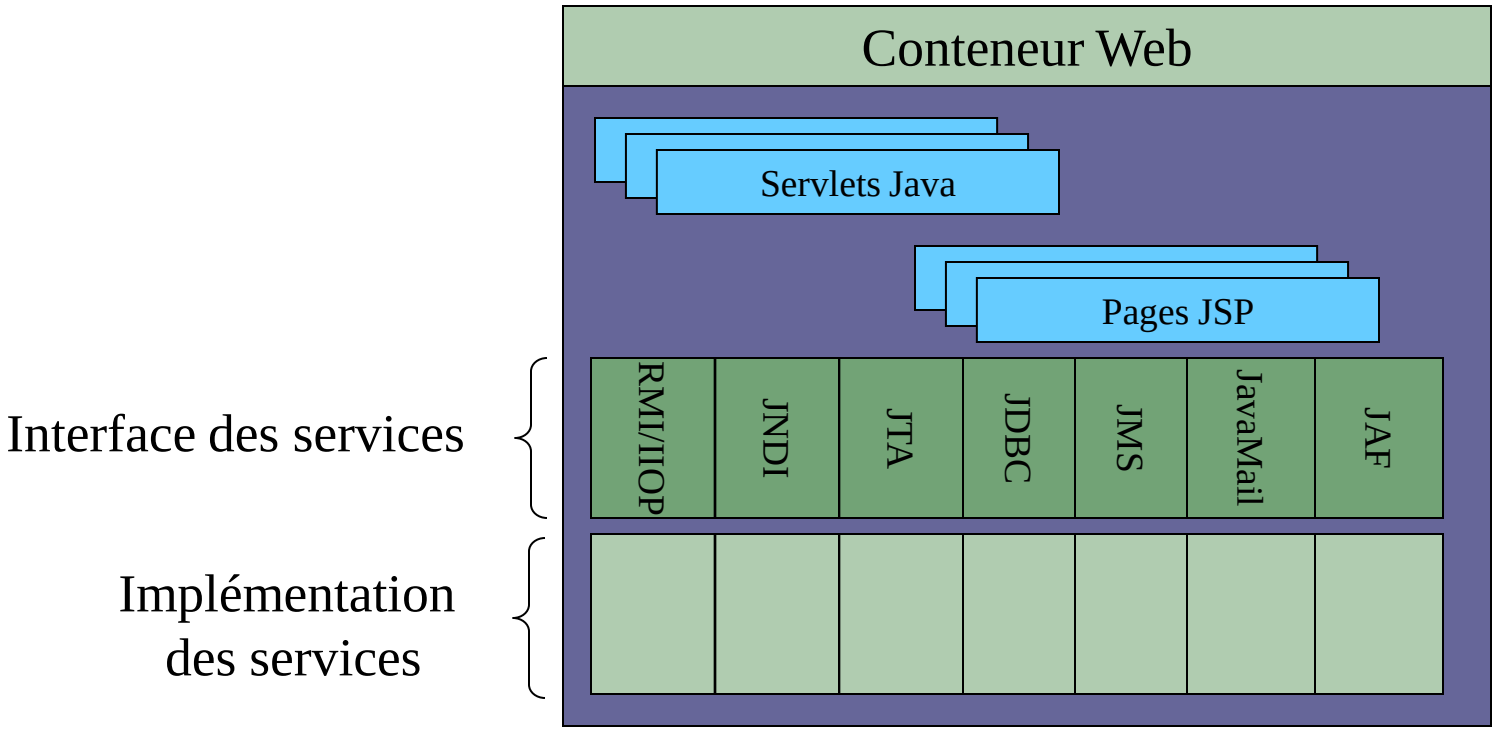
Deruelle Laurent

Le container Web

- Il fournit un environnement pour le développement, le déploiement et la gestion de l 'exécution des *Servlets* et des *JSP*.
- Les Servlets et les JSP sont regroupés dans des unités de déploiement baptisés *applications Web (webapp)*.
- Ils implémentent la logique de présentation d 'une application.

Architecture J2EE – Conteneurs Web

- API de services de conteneurs



Le container EJB

- Environnement pour le développement, le déploiement et la gestion de l 'exécution des *beans enterprise*.
- Un *bean enterprise* est un composant qui implémente un processus et une entité métier de l 'Entreprise.

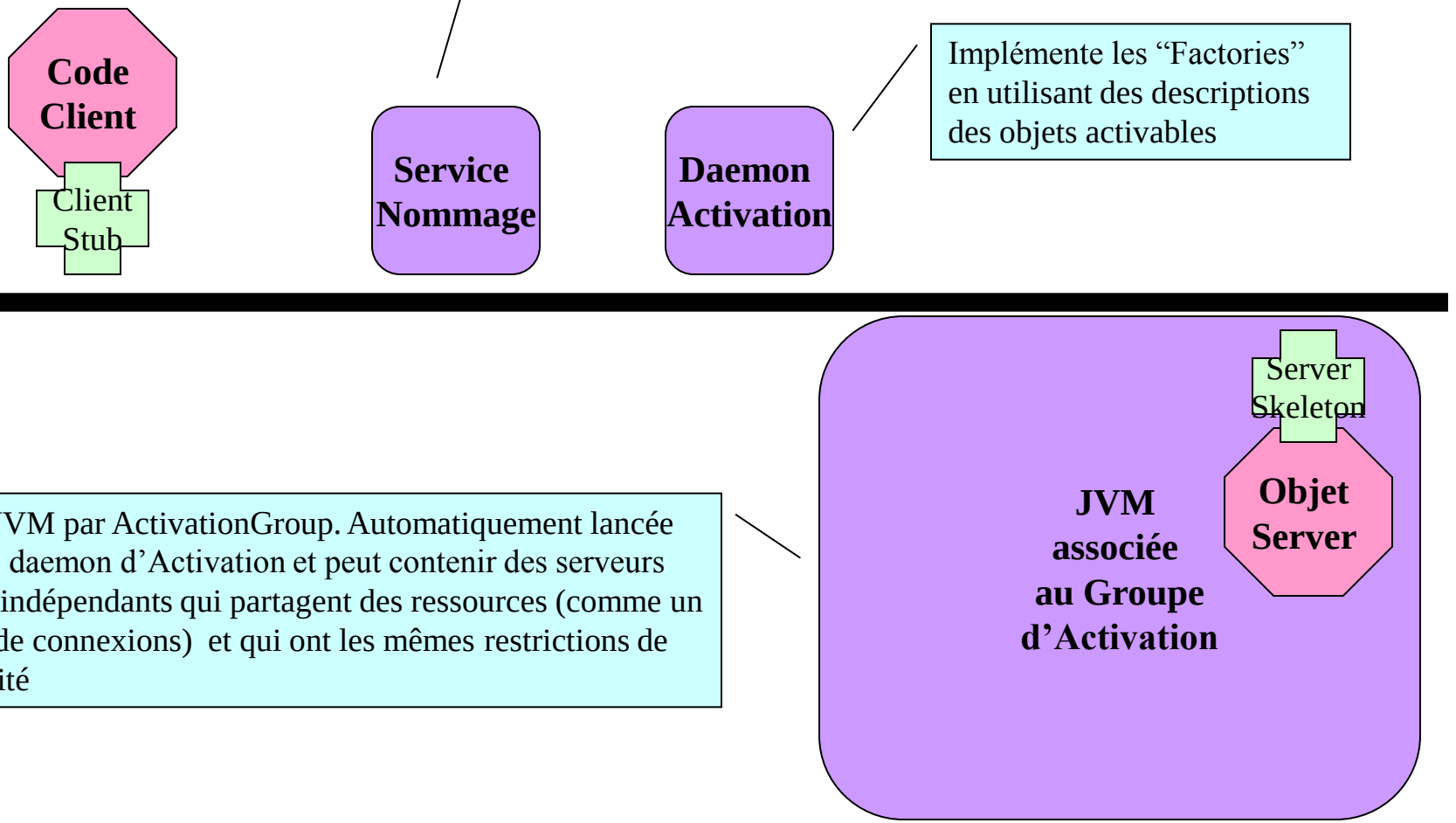
Architecture J2EE - Conteneurs

- Quelques services du conteneur
  - Gestion de la durée de vie des composants applicatifs
    - Cette gestion implique la création de nouvelles instances de composants applicatifs ainsi que le pooling et la destruction de ces composants lorsque les instances ne sont plus nécessaires
  - Pooling de ressources
    - Les conteneurs peuvent à l'occasion mettre en œuvre le rassemblement des ressources, sous la forme, par exemple, de pooling d'objets ou de pooling de connections
  - Peuplement de l'espace de noms JNDI avec les objets nécessaires à l'utilisation des API de services des conteneurs
  - Clustering sur plusieurs machines
    - Répartition de charge ou "Load Balancing"
  - Sécurité
  - ...

Les conteneurs *via* Java RMI

Structure de répertoire qui permet d'obtenir des notions de transparence de la localisation des objets -- le code client utilise un nom pour trouver le serveur

Implémente les "Factories" en utilisant des descriptions des objets activables



Une JVM par ActivationGroup. Automatiquement lancée par le daemon d'Activation et peut contenir des serveurs semi-indépendants qui partagent des ressources (comme un pool de connexions) et qui ont les mêmes restrictions de sécurité

Deruelle Laurent

Les containers *via* CORBA

- Tous les containers comportent un ORB (*Object Request Broker*) compatible CORBA.
- Le protocole d 'échange entre les containers EJB de nombreux éditeurs repose sur CORBA, incluant RMI-IIOP et le service de transaction OTS.
- Certains serveurs d'applications peuvent ainsi autoriser :
  - l'accès à des objets C++ (COM+/DCOM ou standard) fonctionnant sur des machines hétérogènes.
  - La connexion de différents serveurs d'applications à bus CORBA hétérogènes et distribués

Les modèles de composants

- Un composant est un objet qui adhère à un modèle :
  - Support d'un ensemble de méthodes normées pour :
    - être exécuté dans un serveur d'applications,
    - être intégré dans un AGL en phase de développement, grâce aux méthodes auto descriptives (introspection).
  - Regroupement des méthodes normées dans des interfaces :
    - Le composant donne un descriptif de ses services,
    - La description des services est indépendante de l'implémentation.

Les modèles de composants

- La structure du composant doit permettre un réel assemblage avec d'autres composant, sans se soucier des implémentations.
- Le modèle de composants apporte un ensemble d'objets et de méthodes afin de mieux gérer les aspects transactionnels et de sécurité.

Les modèles de composants

- Deux modèles (architectures) de composants se partagent le marché :
  - Le modèle COM (Component Object Model, 1996) de Microsoft, se veut indépendant du langage (VB, C++, C#, Visual J++, etc.)
  - Le modèle EJB (Enterprise JavaBean, 1998), spécification multi éditeur (Sun et IBM à l'origine), se veut multi plateforme et uniquement en langage Java.

Le modèle COM

- Le modèle COM définit un standard pour la communication des objets : assemblage de composants multi éditeurs et multi langages.
- Il désigne les services de base du *broker* d'objet.
- La technologie ActiveX a été développée par dessus COM, pour désigner :
  - les composants destinés à Internet,
  - des contrôles graphiques (OCX),
  - et permettre le pilotage d'application.

Le modèle COM+

- Le modèle COM+ est une évolution de COM à partir de Windows 2000.
- Il intègre de nouvelles fonctionnalités :
  - La gestion d'événement en mode *publish* et *subscribe*,
  - des communications asynchrones et sûres entre composants (MSMQ : Message Queuing 3.0),
  - Les transactions distribuées (MTS).

Architecture de composants COM/COM+

- Du point de vue architecture de déploiement, les composants métiers peuvent être sollicités par :
  - des clients lourds *via* :
    - le protocole DCOM,
    - Le protocole HTTP, au moyen d'IIS.
  - Des clients légers, tels que des pages HTML *via* les ASP.

Le modèle EJB

- Le modèle Enterprise JavaBeans est basé sur le concept *Write Once, Run Everywhere* pour les serveurs.
- Le modèle EJB repose sur l'architecture en couches suivante :
  - L'EJB Server contient l'EJB Container et lui fournit les services de bas niveau.
  - L'EJB Container est l'environnement d'exécution des composants Enterprise JavaBeans (interface entre le bean et l'extérieur).
  - Les clients ne se connectent pas directement au bean, mais à une représentation fournie par le conteneur. Celui-ci

Les services du container EJB

- L'EJB Container est responsable de la fourniture de services aux beans, quelque soit leurs implémentations :
  - Le support du mode transactionnel : spécifié lors de la déclaration du bean sans ajout de codes. La granularité pouvant descendre au niveau de la méthode.
  - La gestion des multiples instances : les EJB sont développés de façon mono-client et exécutée en mode multi-clients:
    - gestion de pool d'instances,
    - gestion de cache,
    - optimisation des accès ressources et données, *etc.*
  - La persistance (obligatoire dans la spécification EJB 2.0).
  - La sécurité par les ACL (Access Control List) au niveau du bean ou pour chaque méthode.
  - Gestion de versions et administration des EJBs.

Les beans enterprise du container EJB

- Un Bean Enterprise (EJB ou beans) est un composant chargé des opérations métiers de l 'application.
- L 'architecture est découpée en quatre tâches:
  - le développement du bean enterprise,
  - l 'assemblage de l'EJB (JAR),
  - le déploiement de l'EJB dans un environnement d 'exécution,
  - l 'administration et la configuration de l'EJB,

Les interactions entre composants

- Un bean encapsule une partie de la logique métier de l'application et des règles business. Ils rendent des services fonctionnels.
- Il communique généralement avec des gestionnaires de ressources (SGBD).
- Les clients des beans sont :
  - d 'autres beans,
  - des appplications Web, des servlets, des java beans,
  - des applications classiques.

Les règles business

- Les applications à base d'EJBs organisent les règles business en composants :
  - une entité business représente les informations conservées par l 'Entreprise
  - un processus business définit les interactions d'un utilisateur avec des entités business.
- Les règles business peuvent être extraites et placées dans un moteur de règles (système expert, etc.), puis manipulées *via* un EJB : nouvelles tendances.

Deruelle Laurent

L'entité business

- Elle possède un état, conservé en permanence (SGBD), modifié généralement par les processus business.
- Exemple:
  - Une entité Commande encapsulera les données des commandes d'un client avec ses règles business (i.e. formatage du N° de commande, vérification du compte du client, etc.)

Deruelle Laurent

Le processus business

- Il modifie l 'état des entités business, et possède son propre état, souvent provisoire.
- Un processus business est dédié à un acteur (utilisateur ou programme) qui engage une conversation avec le système :

  processus business conversationnel

- Exemple :

  une personne retirant de l 'argent à un distributeur

Les types de beans Enterprise

- Les entités et processus business sont implémentés au choix par trois types de beans :
  - les beans entité : créés pour vivre longtemps et être partagés par plusieurs clients,
  - les beans session : créés en réponse aux requêtes d'un seul client.
  - les beans orientés messages
- Les EJBs orientés messages sont une classe à part des EJBs et sont définis en relation avec l'API Java Message Service

Cycle de vie d'un EJB

- Définir le type de l'EJB :
  - Un EJB session,
  - Un EJB Entité
  - Un EJB message
- Développer le bean
  - Ecrire l'interface Home et l'interface Remote
  - Implémenter les services du bean dans une classe
- Déployer le bean sur un serveur d'applications
  - Créer une description du déploiement (souvent en XML)
  - Nommer l'EJB (souvent un JNDI name)
  - Assembler l'EJB dans un fichier jar (+ librairies, + classes utilitaires)
  - Utiliser l'outil de déploiement du serveur d'applications
- Attendre que l'EJB soit sollicité par une requête.

Deruelle Laurent

Les beans et les transactions

- Dans une application J2EE, on utilise des transactions pour :
  - combiner l'envoi et la réception de messages (JMS),
  - Effectuer des mises à jours de bases de données et
  - Réaliser d'autres opérations de gestion de ressources (EAI).
- Ces ressources peuvent être accédées à partir de multiples composants d'application à l'intérieur d'une même transaction.
- Par exemple:

  une servlet peut démarrer une transaction pour accéder à de multiples bases de données, invoquer un enterprise bean qui envoi un message JMS, invoquer un autre enterprise bean pour modifier un ERP en utilisant l'

Les beans et les transactions

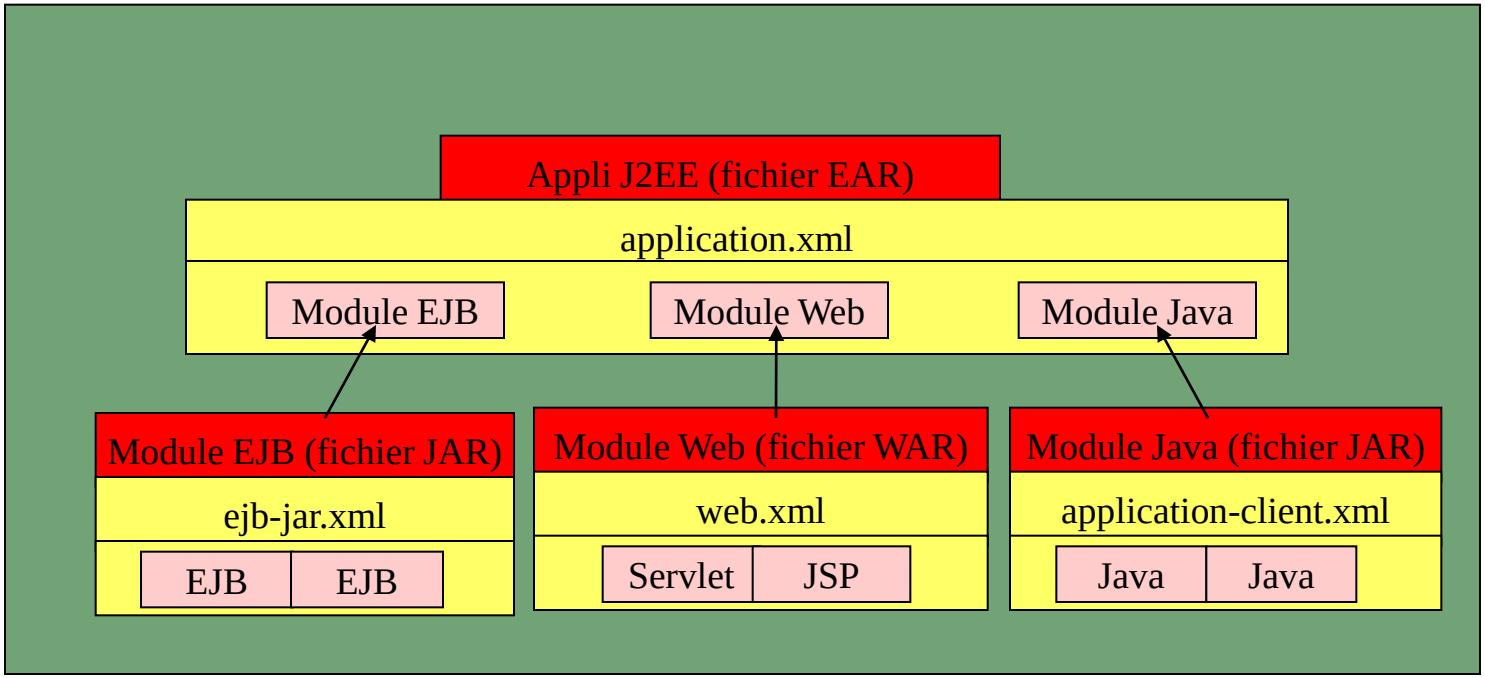
- Les transactions distribuées peuvent être de deux types :
  - **Container-managed transactions.** Le conteneur EJB contrôle l'intégrité de vos transactions sans que vous deviez effectuer un commit ou un rollback.
    - Les CMT sont recommandées pour les applications J2EE qui utilisent JMS.
    - Vous pouvez spécifier des attributs de transaction pour les méthodes des beans.
    - Utiliser l'attribut Required pour s'assurer qu'une méthode fait partie d'une transaction.
    - Lorsqu'une transaction est en cours et qu'une méthode est appelée, celle-ci sera incluse dans la transaction; si aucune transaction n'est en cours, alors une nouvelle transaction sera démarrée avant l'appel de la méthode et sera validée (commit) lorsque la méthode sera terminée.
  - **Bean-managed transactions.** Elles permettent au bean de contrôler finement les transactions via l'interface *javax.transaction.UserTransaction*, permettant d'utiliser ses propres méthodes de commit et de rollback afin de délimiter les frontières des transactions.

Développer des applications J2EE

- Constitution de composants applicatifs en modules
  - Un module sert a empaqueter un ou plusieurs composants du même type
  - L'application J2EE est une archive **EAR** contenant le descripteur de déploiement (**application.xml**), les modules Web et EJB
  - modules Web
    - Servlets, JSP, TagLibs, JARs, HTML, XML, Images...
    - Empaqueté dans un fichier d'archive web, **WAR**
    - Un WAR s'apparente a un JAR avec en plus un répertoire WEB-INF contenant le descripteur de déploiement **web.xml**
  - modules EJB
    - pour les EJB et leurs descripteurs de déploiement (**ejb-jar.xml**) contenus dans une archive **JAR**
  - modules Java
    - pour les clients Java, également une archive **JAR** avec le descripteur de déploiement **application-client.xml**

Développer des applications J2EE

- Constitution de modules en application
  - Niveau le plus accompli : celui des applications
  - Appli J2EE = ensemble de modules places dans un fichier EAR (Entreprise Archive)



Développer des applications J2EE

- Déploiement d'applications
  - Le déploiement consiste à installer et à personnaliser des modules empaquetés sur une plate-forme J2EE
  - Deux étapes :
    - Préparation de l'appli (recopie des fichiers JAR, WAR, EAR..., génération des classes au moyen du conteneur, puis installation de l'appli sur le serveur
    - Configuration de l'application en utilisant les informations spécifiques au serveur d'applications
      - Création de sources de données, fabriques de connexion...

Le serveur J2EE Sun

- Télécharger le Java 2 Standard Edition à partir du site de Sun, pour bénéficier de Java 1.4
- Télécharger le Java 2 Enterprise Edition à partir du site de Sun, pour installer le serveur J2EE.
- Configurer correctement les paramètres du serveur J2EE Sun
- Lancer l'outil j2ee.bat –verbose pour démarrer le serveur
- Développer une application J2EE avec un IDE
- Lancer l'outil deploytool.bat et effectuer l'assemblage de l'application
- Tester l'application :
  - À partir d'une application cliente (mode intranet)
  - À partir d'un navigateur web (mode Internet / Extranet)

Configuration du serveur J2EE Sun MicroSystem

Platform	Variable Name	Values
Microsoft Windows	%JAVA_HOME%	Directory in which the Java™ 2 SDK, Standard Edition, version 1.3.1, is installed
	%J2EE_HOME%	Directory in which the J2EE SDK 1.3.1 is installed, usually `C:\j2sdkee1.3.1`

	%CLASSPATH%	Include the following:
		`.;%J2EE_HOME%\lib\j2ee.jar;`
		`%J2EE_HOME%\lib\locale`

	%PATH%	Include `%J2EE_HOME%\bin`
UNIX	$JAVA_HOME	Directory in which the Java 2 SDK, Standard Edition, version 1.3.1, is installed
	$J2EE_HOME	Directory in which the J2EE SDK 1.3.1 is installed, usually `$HOME/j2sdkee1.3.1`

	$CLASSPATH	Include the following:
		`.:$J2EE_HOME/lib/j2ee.jar:`
		`$J2EE_HOME/lib/locale`

	$PATH	Include `$J2EE_HOME/bin`

Deruelle Laurent